HAL
open science

Optimisation des codes en métrique rang pour les systèmes de communication sans fil

Imad El Qachchach

To cite this version:

Imad El Qachchach. Optimisation des codes en métrique rang pour les systèmes de communication sans fil. Traitement du signal et de l'image [eess.SP]. Université de Limoges, 2019. Français. NNT : 2019LIMO0024 . tel-02872876

HAL Id: tel-02872876

https://theses.hal.science/tel-02872876

Submitted on 18 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LIMOGES

ÉCOLE DOCTORALE 521 :

Sciences et Ingénierie pour l'Information, Mathématiques

XLIM - Systèmes & réseaux intelligents

Année : 2019

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Sciences et technologies de l'information et de la communication

présentée et soutenue par

IMAD EL QACHCHACH

le 17 Juin 2019

Optimisation des codes en métrique rang pour les systèmes de communication sans fil

Thèse dirigée par Jean-Pierre Cances et Oussama Habachi

JURY :

Rapporteurs :

Catherinne Doulliard	Professeur, Télécom Bretagne
Maryline Hélard	Professeur, INSA Rennes

Examinateurs :

Antoine Berthet	Professeur, Centrale Supélec
Jean Pierre Cances	Professeur, ENSIL Limoges
Oussama Habachi	Maître de Conférence, Université de Limoges
Phillipe Gaborit	Professeur, Université de Limoges
Vahid Meghdadi	Professeur, ENSIL Limoges
Emmanuel Radoi	Professeur, Université de Bretagne Occidentale

Table des matières

Table des matières 3

Table des acronymes 5

1 Introduction générale 7
1.1 Le codage réseau 8
1.2 Les codes correcteurs d'erreurs pour des réseaux non cohérents 13
1.3 Motivations, objectifs et contributions de la thèse 13
1.4 Structure de la thèse 14

2 Introduction aux codes en métrique rang 17
2.1 Rappels sur les corps finis 17
2.2 Les q-polynômes 18
2.2.1 L'anneau des q-polynômes 19
2.2.2 La division euclidienne à droite 20
2.2.3 Le plus grand diviseur commun à droite 21
2.3 Présentation des codes et leurs propriétés 22
2.3.1 Codes en bloc linéaires 23
2.3.2 Le principe de décodage 24
2.4 Rappel sur la métrique rang 26
2.4.1 Définitions et propriétés 26
2.4.2 Bornes sur les codes en métrique rang 27
2.5 Les codes Gabidulin 28
2.5.1 Définition et propriétés 28
2.5.2 Décodage des codes Gabidulin 30
2.5.3 Décodage des erreurs avec l'algorithme de Berlekamp-Massey modifié 32
2.6 Conclusion 36

3 Les codes LRPC 37
3.1 Les codes LRPC 38
3.1.1 Le décodage des codes LRPC 38
3.1.2 Exactitude de l'algorithme de décodage 41
3.1.3 La probabilité de succès de décodage 41
3.1.4 La complexité de décodage 41
3.2 Les Codes LRPC matriciels 42
3.2.1 Définition et propriétés 42

3.2.2 Le décodage des codes LRPC matriciels 43
3.2.3 Comparaison des codes LRPC matriciels et des codes Gabidulin . . 45
3.2.4 Les codes LRPC matriciels et la borne de Gilbert-Varshamov 48
3.3 Les codes LRPC étendus 49
3.3.1 Définitions et propriétés 49
3.3.2 Le décodage des codes LRPC étendus 49
3.3.3 La probabilité de décodage des codes LRPC étendus 50
3.4 Conclusion 54

4 Codage réseau 55
4.1 Le codage réseau 56
4.1.1 Le codage réseau linéaire 58
4.1.2 Approche algébrique 60
4.1.3 Exemple de décodage du codage réseau 62
4.2 Le codage réseau linéaire aléatoire 64
4.2.1 L'approche générale 65
4.2.2 Réseau à un seul saut 66
4.2.3 Réseau multi-sauts 70
4.3 Modèle de collecte de données en présence d'erreurs 72
4.3.1 Formulation du problème 74
4.3.2 Modèle proposé 77
4.4 Résultats de simulation du modèle proposé 78
4.4.1 Paramètres de simulation 78
4.4.2 Résultats de simulation 79
4.4.3 Comparaison de la complexité 80
4.5 Conclusion 81

5 Codage réseau multi-sources 83
5.1 Description du code 84
5.1.1 Codage réseau cohérent 86
5.1.2 Codage réseau non cohérent 87
5.1.3 Décodage des codes en métrique rang 88
5.1.4 Décodage des codes LRPC 90
5.2 Description du modèle 92
5.3 Résultats numériques 96
5.4 Conclusion 99

6 Conclusion et perspectives 101

Bibliographie 105

Table des acronymes

AWGN	Additive White Gaussian Noise
BPSK	Binary Phase-Shift Keying
BS	Base Station
CC	Code Convolutif
CRC	Cyclic Redundancy Check
IoT	Internet of Things
IoV	Internet of Vehicles
LRPC	Low Rank Parity Check
MRD	Maximum Rank Distance
NC	Network Coding
RC	Rank Code
RGCD	Right Greatest Common Divisor
RLNC	Random Linear Network Coding
RS	Reed Solomon
SF	Store and Forward
WSN	Wireless Sensor Network

Chapitre 1

Introduction générale

Les systèmes de télécommunications ont connu un changement de paradigme au cours des deux dernières décennies avec l'arrivée de l'Internet des objets (IoT). Les réseaux sans fil supportent un trafic de plus en plus important en raison de la croissance spectaculaire des appareils mobiles, du développement de diverses applications ainsi que la mise en œuvre de l'IoT. Par conséquent, les nouveaux systèmes de communication sans fil ont mis en avant des indicateurs de performance divers et souvent contradictoires, tels que la capacité élevée, le faible temps de latence, la fiabilité, la mobilité élevée et la connectivité massive. Nombreuses sont les applications qui nécessitent l'optimisation de l'un ou plusieurs de ces paramètres. Nous pouvons citer comme exemple l'Internet des Véhicules (IoV) qui est une application particulière de l'IoT où les véhicules sont interconnectés afin d'obtenir un trafic plus fluide [Fan+14]. Dans cet exemple d'application, les véhicules se comportent comme des capteurs, leur rôle essentiel est de collecter les données et de les envoyer vers un système de transport intelligent. Les informations collectées sont des données d'environnement ou des données en provenance d'autres véhicules, et elles sont utilisées pour une gestion du trafic en temps réel. Dans ce cas de figure, l'objectif est d'avoir un faible temps de latence pour avoir un traitement rapide de l'information. De plus, les véhicules ont une mobilité élevée, ce qui ajoute des contraintes supplémentaires au système d'optimisation.

Dans d'autres applications, y compris l'IoV, il y a une nécessité de collecter les données, depuis l'environnement physique, par des nœuds sources vers la destination. La collecte de données se fait par le biais des nœuds, appelés capteurs, qui effectuent la mesure physique et envoient ces informations vers des points de collecte. La mise en place des réseaux de capteurs sans fil (WSNs), permet de l'exploration de l'environnement par les capteurs et contribue ainsi au développement de l'IoT dans de nombreux domaines : agriculture, transport, gestion de l'énergie, etc.

La mise en place des WSNs est donc devenue un enjeu économique et sociétal particulièrement important dans les grands centres urbains. Les WSNs sont une famille des réseaux Low Power Wireless Personal Area Network (LoWPAN). Ces réseaux se constituent d'un grand nombre de composants, appelés capteurs, qui communiquent entre eux à travers un canal sans fil à faible portée. Ces capteurs collectent les données

physiques et coopèrent parfois dans le but de transmettre ces information vers une ou plusieurs destination de collecte mais aussi afin d'interagir avec leur environnement comme des actionneurs [Mon+07].

L'architecture des réseaux a été pensée dans l'objectif de limiter le coût de fabrication et de permettre un déploiement à large échelle. Ils fonctionnent avec une puissance de calcul faible, une mémoire limitées et disposent d'une source énergétique limitée [Aky+02]. Ainsi, ces nœuds à bas coût sont petits, ce qui rend leur intégration facile et massive dans différents environnements comme les smart cities. Les applications basées sur les WSNs proposent un déploiement aléatoire d'un très grand nombre de capteurs.

Suivant l'application envisagée, les capteurs sont déployés soit dans des environnements hostiles ou dans des environnements faciles d'accès tels que les centre-villes. Le déploiement peut se faire d'une manière aléatoire (exemple des capteurs sismiques) ou suivant un plan défini. A cause du grand nombre de nœuds déployés ainsi qu'à la difficulté d'accéder à certains d'entre de ces environnements, le changement des piles d'alimentation est parfois impraticable ou très coûteux. Les nœuds doivent donc maximiser leur durée de vie. En effet, la durée de vie moyenne espérée est d'environ 10 ans. Les transmissions ainsi que les protocoles utilisés par les capteurs du réseau doivent donc être à basse consommation. Cependant, la contrainte énergétique n'est pas le seul problème qu'on peut rencontrer dans les réseaux de capteurs, le problème du routage est aussi un défi important dans la mise en œuvre de ces réseaux de capteurs [Ahl+00].

Le routage multisaut, appelé *store and forward* en anglais, est le moyen traditionnel d'acheminement de données dans un réseau. Dans ce protocole, les nœuds intermédiaires retransmettent simplement les paquets reçus vers leur destination. En revanche, comme la topologie du réseau est dynamique et les liens ont des délais différents, la transmission d'un grand nombre de paquets par les nœuds intermédiaires crée des répliques du paquet à plusieurs endroits à la fois dans le réseau, ce qui peut conduire à la saturation de ce dernier. Puisque l'ordre des paquets sources n'est pas connu, dans la plupart des cas la destination reçoit plusieurs copies d'un même paquet source. De plus, les paquets arrivent de façon désordonnée rendant ainsi la récupération des messages très délicate. En conséquence, le délai de transmission dans le réseau est augmenté et le débit est réduit. Afin de remédier à ce problème, de nouvelles approches de transmission d'information ont été proposées pour les WSNs. Ces solutions utilisent des combinaisons linéaires de données en provenance de différentes source afin d'atteindre la limite théorique d'utilisation de la bande passante, et cela que ce soit dans des connections réseau unicast ou des connections réseau multicast [YZ99] ; [Ahl+00].

1.1 Le codage réseau

Depuis sa première publication par Ahlswede et al. dans l'article [Ahl+00], le codage réseau (NC) est devenue une discipline indispensable de la télécommuni-

cation. Ce nouveau concept a remplacé les méthodes du routage classique, ce qui a ouvert des nouvelles pistes de recherche dans les domaines du codage canal, du traitement de signal et des communications filaires et/ou sans fil. Utilisés d'abord dans les réseaux filaires en multicast, et adaptés ensuite aux réseaux unicast puis aux réseaux sans fil, les principes de cette théorie ont complètement révolutionné la manière dont les réseaux de communications sont perçus aujourd'hui. Cette nouvelle technique considère que les paquets ne sont plus des entités immuables mais qu'on peut les combiner, les recombiner ou ne pas les transmettre par différents nœuds. Divers domaines sont concernés par l'application de la technique du codage réseau tels que les communications satellitaires [YZ99], les systèmes de fichiers distribué [Ace+05], la sécurité ou encore les réseaux d'information quantiques [FS+08].

Par ailleurs, le codage réseau est une technique efficace de la transmission des données dans un canal à effacement. Cela est dû principalement à sa propriété d'exiger des acquittements et au faible nombre de paquets circulant dans le réseau.

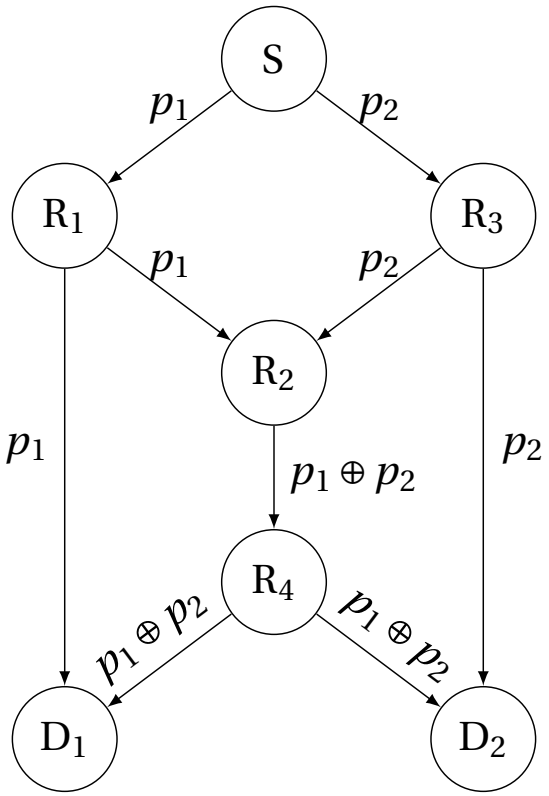


FIGURE 1 – Exemple de codage réseau dans un réseau de capteurs à une source et deux destination

On peut distinguer deux types de codage réseau, le codage pour des réseaux cohérents et le codage pour des réseaux non cohérents. Dans le cas du codage pour des réseaux cohérents, la topologie du réseau ainsi que le codage réseau utilisé sont supposés connus par les nœuds destinations. Considérons un simple exemple de réseau, appelé *réseau papillon*, dans la Figure 1. Ce réseau comprend une source S,

deux destinations D_1 et D_2 et quatre nœuds intermédiaires. La source génère deux paquets p_1 et p_2 et les transmet aux destinations qui demandent les deux paquets. Le nœuds R_2 reçoit les paquets p_1 et p_2. Si aucun codage n'est utilisé, le nœud R_2 envoie l'un des deux paquets et conserve l'autre dans la mémoire. Ce paquet sera envoyé dans le créneau temporel suivant. Donc, pour que les deux destinations reçoivent tous les paquets, il faut réaliser deux transmissions. D'autre part, si on utilise le codage réseau, le nœud R_2 envoie la somme des deux paquets comme on peut remarquer dans la Figure 1. Ainsi, la destination t_1 reçoit p_1 et $p_1 \oplus p_2$, puis elle récupère le paquet p_2 en faisant $p_1 \oplus (p_1 \oplus p_2)$. Dans ce cas, le codage réseau augmente le débit de communication dans le réseau. Plus précisément, le codage réseau permet de se rapprocher de la capacité maximale du réseau.

Contrairement à l'exemple traité dans le paragraphe précédent, un réseau réel est plus grand et plus complexe que le réseau papillon. En plus, un réseau de capteurs sans fil ne possède pas d'entité centrale capable de chercher des combinaisons optimales pour les nœuds intermédiaires et d'envoyer cette solution aux nœuds destinations pour la récupération des paquets sources. Ce contexte s'applique parfaitement au cas des réseaux non cohérents. En effet, deux contributions, par Ho, Medard et Koetter, ont transformé le codage réseau d'une approche purement théorique à une solution réelle, le codage réseau linéaire [Ho+03], qui est étendu dans [Ho+06] au codage réseau linéaire aléatoire (RLNC). Ce dernier est considéré comme la technique la plus performante dans les réseaux de capteurs non cohérents qui permet d'avoir une implémentation très simple et relativement indépendante de la structure du réseau considéré. Dans ce cas, les paquets sont vus comme des vecteurs de longueur N sur un corps fini $\mathbb{F}_q$. Les nœuds intermédiaires choisissent ainsi aléatoirement une ou plusieurs combinaisons linéaires des paquets qu'ils ont reçus. Bien que le codage RLNC ne garantisse pas que toute l'information soit correctement transmise en un temps minimal, des théorèmes que l'on peut trouver dans [Ho+06] montrent que la probabilité de recevoir le message en entier en temps minimal tend exponentiellement vers 1 en fonction de la taille de corps utilisé, ce qui fait que la capacité maximale du réseau est atteinte. En revanche, comme la complexité de combinaison et de décodage augmente avec la taille du corps, des corps du type $\mathbb{F}_q$ où q est une puissance de 2 sont proposés en général. La source doit encoder ses messages en vecteurs linéairement indépendants. Contrairement à l'approche typiquement proposée, nous considérons des vecteurs non nuls et dont la première coordonnée non nulle est égale à 1. Cette méthode sera détaillée dans le **chapitre 4**. Afin d'encoder ses messages, la source ajoute une entête formée par la matrice identité concaténée avec les paquets à transmettre. Les messages ainsi encodés en k paquets linéairement indépendants sur $\mathbb{F}_q$ sont transmis dans le réseau. Puisque les combinaisons linéaires opérées par les nœuds intermédiaires ne modifient pas le sous espace engendré par les lignes de la matrice transmise, le codage RLNC est vu comme la transmission d'un sous-espace linéaire de dimension k sur $\mathbb{F}_q$ [KK08] ; [SKK08]. Plus précisément, la source envoie la matrice encodée et chaque destination recevra une matrice de la concaténation des données codées et d'une matrice contenant les opérations qui se sont passées au sein des nœuds intermédiaires. Si la matrice reçue est de rang k, la destination récupère les paquets transmis en faisant une simple résolution d'un système linéaire d'équations.

L'utilisation du codage réseau aléatoire dans un réseau de capteur, dans le cas où il ne permet pas d'augmenter le débit, permet de simplifier la structure du réseau. Cette simplification vient du fait que toute forme d'intelligence est supprimée du réseau. Chacun des nœuds intermédiaires choisit les combinaisons uniformément et indépendamment des autres nœuds, ce qui permet de diminuer le coût de fabrication des nœuds de ce réseau.

La propagation des erreurs

Par ailleurs, le codage réseau n'est pas une technique sans inconvénient. En utilisant cette technique dans un réseau, on risque de perdre tous les paquets sources. En effet, le codage réseau est très sensible à la propagation des erreurs de transmission. Des paquets corrompus, intentionnellement ou à cause d'un utilisateur malveillant, peuvent être combinés avec d'autres paquets intègres ce qui entraine la perte de la totalité des paquets sources. Plus précisément, les combinaisons effectuées par chaque nœud entrainent une contamination progressive de l'ensemble des paquets non erronés par les paquets erronés, rendant le décodage impossible au niveau du récepteur (voire Figure 2).

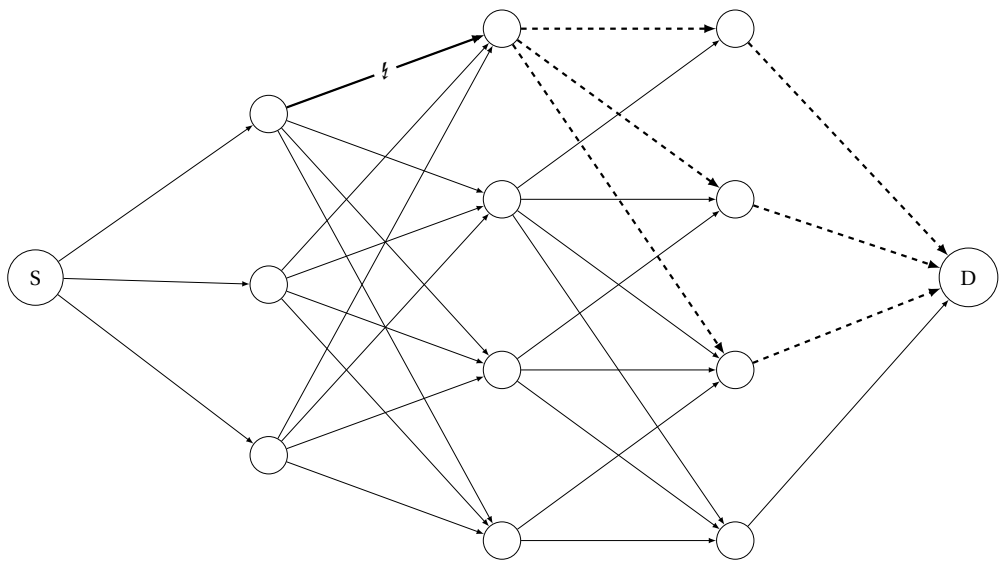


FIGURE 2 – Exemple de propagation d'erreur dans un réseau de capteur lors de l'utilisation du codage réseau.

Ce scénario donne un avantage au routage par rapport au codage réseau, où seulement les paquets contaminés ou non reçus sont retransmis. De plus, si le problème persiste, l'algorithme du routage pourra chercher un autre chemin. Pour remédier à la propagation d'erreurs, nous pouvons penser à combiner le codage réseau avec des algorithme de routage. Un algorithme de routage adaptatif permet de choisir un autre chemin lorsqu'il y a eu des paquets corrompus détectés. Par ailleurs, même

en absence d'erreurs, les pertes de paquets conduisent à la réception d'un nombre insuffisant de paquets par la destination, et rendent l'exploitation des paquets déjà reçus impossible car le système d'équations est incomplet.

En conséquence du phénomène de propagation d'erreurs, les paquets dans le réseau sont perdus même dans le cas d'utilisation d'un code correcteur d'erreur classique pour la protection des données de bout-en-bout. Les codes correcteurs classiques, qui sont des codes dans la métrique de Hamming, sont utilisés pour la correction d'erreurs où la matrice de représentation est de faible densité, i.e. *la matrice erreur* contient un petit nombre d'éléments non nuls. En effet, même dans le cas où la matrice erreur a une faible densité, en combinant ses lignes, la densité à la destination devient aléatoire et proche de $\frac{1}{2}$ si chaque colonne de la matrice erreur contient un élément non nul. Dans ce cas, la destination ne pourra pas récupérer les paquets source. Par conséquent, une nouvelle technique de codage d'erreur est nécessaire pour la récupération de ce type d'erreur.

Les scénarios d'erreurs, que nous pouvons considérer dans un réseau, sont nombreux. Dans le cas où les erreurs sont peu fréquentes dans le réseau, on peut utiliser un outil de détection d'erreur comme le contrôle de redondance cyclique (CRC), dans le but de rejeter les paquets corrompus. Cela conduit à avoir des effacements à la place des erreurs à la réception. Cependant, cette solution a un inconvénient majeur : le contrôle de chacun des paquets reçus par les nœuds intermédiaires conduit à l'augmentation du délai de transmission à cause du grand nombre de paquets reçus. Après avoir vérifié les paquets reçus, le nœud forme des combinaisons des paquets non corrompus et les réencode avec l'algorithme de contrôle CRC. Ce traitement implique une augmentation du coût de fabrication à cause de l'implémentation des algorithmes de codage et de contrôle. De plus, le rejet d'un grand nombre de paquets conduit à la réception d'un nombre insuffisant de paquets à la destination. Les codes correcteurs d'erreur puissants, comme les codes Reed-Solomon, exigent un grand nombre d'opérations et ont un grand coût de fabrication rendant leur utilisation quasiment impossible dans un réseau de capteur. Les codes correcteurs d'erreurs itératifs, comme les codes LDPC, nécessitent des tailles de paquets très grandes, et en conséquence le nœud ne reçoit pas tous les fragments de paquets codés. L'utilisation d'un code correcteur d'erreur à bas coût dans chaque nœud réduit significativement l'effet du bruit de fond et cela permet d'éviter la perte totale des données source. Cette technique s'appelle *decode and forward*. Nous présentons en détail ce modèle en utilisant l'exemple du code convolutif dans le chapitre 4 de cette thèse.

Dans un autre scénario, nous étudions la présence d'un utilisateur malveillant dans le réseau. Ce dernier a la capacité de générer des paquets du même format que les paquets source(entête + un message codé). Ce format est supposé être connu par tous les nœuds du réseau. Dans le cas où les nœuds intermédiaires utilisent un contrôle CRC, l'adversaire peut également utiliser le contrôle CRC pour la génération des paquets. Dans le cas où l'adversaire ne connait pas le détail de l'entête, il peut utiliser les paquets générés par la source pour fabriquer un nouveau paquet.

1.2 Les codes correcteurs d'erreurs pour des réseaux non cohérents

Les techniques de codage réseau correcteur d'erreurs ont pour objectif de protéger les paquets transmis vis-à-vis des paquets erronés ou des pertes. Ces techniques introduisent un certain niveau de redondance, et sont similaires dans leur principe aux codes correcteurs d'erreurs classiques. Deux familles de codes peuvent être distinguées, les codes en métrique rang et les codes correcteurs d'erreurs, introduits dans [CY02], qui considèrent conjointement le codage réseau et un code correcteur. Ces codes nécessitent une connaissance a priori de l'architecture du réseau et de la manière dont le codage réseau est effectué. Ces résultats sont étendus dans le cadre d'un codage réseau aléatoire dans [Ho+06]. La technique introduite dans [KK08] exploite le fait qu'en absence d'erreurs, le codage réseau aléatoire conserve l'espace vectoriel engendré par les paquets transmis. Dans ce cas, le code est constitué d'un ensemble d'espaces vectoriels. La source transmet des paquets formant la base de l'un de ces espaces, et la destination cherche à retrouver l'espace envoyé par la source. Les codes réseaux robustes obtenus ont des propriétés indépendantes de la structure du réseau.

La recherche dans le domaine des codes correcteurs d'erreur pour le codage réseau a commencé dans les articles [Yeu+05] ; [CY+06] dans lesquels les auteurs remettent en question le problème de correction d'erreurs dans le cas d'un réseau de capteur cohérent. Les auteurs supposent que les destinations ont une connaissance de l'ensemble des opérations qui se sont produites lors de la transmission ainsi que la topologie du réseau. Cela implique qu'on peut remonter jusqu'à la source pour prédire la valeur exacte de l'erreur en utilisant la méthode de décodage par recherche de vecteur erreur de poids minimal ou par maximum de vraisemblance. Il a été démontré dans [Yeu+05] que la capacité de correction d'un code réseau peut atteindre la borne de Singleton.

1.3 Motivations, objectifs et contributions de la thèse

Les techniques de codage réseau correcteurs d'erreurs proposés dans [KK08] et [SKK08], sont très différentes des techniques présentées dans le paragraphe précédent. Ces codes sont efficaces dans n'importe quel type de codage réseau. En effet, le réseau peut être simplifié en un canal qui a des propriétés spécifiques. La propriété la plus importante est la préservation du rang de la matrice erreur lors de la transmission jusqu'à son arrivée à la destination. Lors de la transmission, les paquets peuvent être considérés comme des vecteurs lignes d'une matrice. Dans ce cas, les opérations effectuées dans les nœuds sont des combinaisons des lignes de cette matrice. Ainsi, l'espace généré par les lignes de la matrice construite à la source est conservé. En se basant sur ces observations, Silva, Kschischang et Kötter ont proposé un modèle de code correcteur d'erreur en métrique de sous-espace pour le codage réseau aléatoire. Un code sous-espace est un ensemble de vecteurs de longueur N dans un corps de Galois, chaque mot de code est lui même considéré comme étant un sous-espace vectoriel. Il a été démontré dans [SKK08] que

les codes Gabidulin peuvent être transformés en des codes *Gabidulin relevés* quasi optimaux dans la métrique sous-espace. Pour ce faire, il suffit de concaténer les mots de code, sous leurs formes matricielles, avec une matrice identité. Les codes de Gabidulin sont les analogues en métrique rang des codes de Reed–Solomon en métrique de Hamming. Un code en métrique rang est un ensemble non vide dont la distance entre deux mots de code est le rang de la différence de leurs matrices associés.

Dans ce contexte, cette thèse focalise sur le problème de reconstruction de données émises par les sources dans le cas du codage réseau non cohérent. Nous visons à travers cette thèse à adapter le code Low Rank Parity Check (LRPC) [Gab+13] dans le contexte du codage réseau non cohérent pour les réseaux de capteurs, à une présence possible de trois types d'erreurs, le bruit de fond, les erreurs injectés dans le réseau par un utilisateur malveillant et les effacements qui peuvent être dus aux pannes des nœuds ou aux combinaisons effectuées par les nœuds intermédiaires, qui produise un système d'équations incomplet. Les contributions principales que nous avons apportées à travers ce mémoire sont les suivantes :

— Nous présentons un schéma de codage utilisant les codes LRPC dans le contexte du codage réseau aléatoire pour un réseau avec une seule source, puis multi-sources et nous dérivons analytiquement la probabilité de décodage des codes LRPC pour les différents scénarios considérés.

— Nous proposons une amélioration des codes LRPC pour qu'ils soient bien adaptés au codage réseau aléatoire et nous comparons leurs performances avec les codes Gabidulin relevés.

— Nous montrons, à travers des simulations, que la probabilité de décodage calculée analytiquement pour les codes LRPC est exacte, et que le schéma proposé est plus performant que le schéma basé sur les codes Gabidulin relevés en ce qui concerne la capacité de correction et la complexité.

— Nous montrons que pour certains paramètres du code, un code LRPC peut asymptotiquement atteindre la borne de Gilbert-Varshamov en métrique rang.

1.4 Structure de la thèse

Cette thèse est structurée comme suit :

— Dans le **chapitre 2**, nous présentons les propriétés des corps finis, de la théorie des codes et des polynômes linéarisés essentielles pour cette thèse. Ensuite, nous introduisons les codes en métrique rang et leurs propriétés. Nous définissons les codes Gabidulin comme une évaluation de polynômes linéarisés et nous présentons leur matrice génératrice ainsi que leur matrice de contrôle. Nous calculons la capacité de correction de ce type de code qui atteint la borne de Singleton et nous présentons leur algorithme de décodage.

— Le **chapitre 3** introduit les codes LRPC et leurs propriétés. Nous présentons un algorithme de décodage efficace pour les erreurs de type rang. Ensuite, nous présentons la famille des codes LRPC matriciels qui est une simplification des codes LRPC. Nous montrons que ces codes peuvent asymptotiquement atteindre la

borne de Gilbert-Varshamov. Dans ce contexte, nous analysons leurs performances en les comparant avec les codes Gabidulin en terme de capacité de correction. Ensuite, nous présentons les Codes LRPC étendus qui sont une généralisation des codes LRPC. Nous présentons leur algorithme de décodage et nous calculons la probabilité d'échec de leur algorithme de décodage. Enfin, nous analysons leurs performances à travers des simulations.

— Dans le **chapitre 4**, nous introduisons le codage réseau ainsi que ses principales propriétés. Nous donnons les conditions sous lesquelles la destination peut récupérer les paquets source. Par la suite, nous considérons un scénario de transmission d'une source vers une destination à travers des nœuds intermédiaires en présence de trois types d'erreurs : le bruit de fond, les erreurs injectés dans le réseau par un utilisateur malveillant et les effacements qui peuvent être dus aux pannes des nœuds ou aux combinaisons effectuées par les nœuds intermédiaires. Nous utilisons dans ce cas une concaténation d'un code LRPC et un code convolutif. Nous dérivons analytiquement la probabilité d'échec de décodage des codes LRPC et nous montrons l'exactitude de de la probabilité calculée à travers des simulations. Nous montrons également que le schéma proposé est plus performant que le schéma existant dans la littérature en terme de capacité de correction.

— Dans le **chapitre 5**, nous considérons un scénario de transmission dans un réseau multi-sources. Nous prouvons que les codes LRPC dans ce cas se comporte mieux que dans le cas d'un réseau à une source grâce à leurs structures. Nous montrons également que les codes LRPC résistent aux effacements contrairement aux codes Gabidulin.

— Le **chapitre 6** conclut le manuscrit. Il résume le contenu de la thèse et ses principales contributions puis présente quelques perspectives.

Chapitre 2

Introduction aux codes en métrique rang

Les codes en métrique rang ont été introduits dans [Del78] pour la correction des erreurs. Delsarte a montré l'existence de la borne de Singleton pour la métrique rang, et il a construit une famille de codes qui atteint cette borne et que l'on appellera par la suite les codes Gabidulin. Dans son article [Gab85], Gabidulin a déduit un algorithme de décodage pour ces codes qui peut décoder jusqu'à leur capacité de correction ainsi que d'autres propriétés pour ces codes. Dans des travaux indépendants, Roth a introduit les codes en métrique rang et a proposé des applications pour la correction des tableaux d'erreurs de type "criss-cross" [Rot91].

Ce chapitre est d'une grand importance dès lors qu'il introduit les propriétés fondamentales des corps finis et décrit les méthodes de construction de ces derniers. Nous introduisons, dans ce chapitre, la notion des codes correcteur d'erreurs. Ensuite, nous donnons une construction des codes Gabidulin, similaire à celle des codes Reed-Solomon, basée sur l'évaluation d'un polynôme sur un vecteur. Dans la dernière partie de ce chapitre, nous présentons un algorithme de décodage, en temps polynomial, pour les codes Gabidulin.

2.1 Rappels sur les corps finis

Dans cette section, nous présentons les notions essentielles sur les corps finis, sans entrer dans les détails des démonstrations, pour aider le lecteur à la compréhension de cette thèse. Le lecteur est invité à lire [LNC83] et [Ber84] pour plus de propriétés et d'exemples, ainsi que les preuves détaillées des théorèmes.

Soit p un nombre premier et F un corps fini. Soit K un sous-corps de F contenant p élément. On appelle alors K le sous-corps premier de F et p la caractéristique de F et de K. De plus, F contient p^m éléments, où, m est le degré de F sur K. Maintenant, nous présentons le théorème principal caractéristique des corps finis [LNC83, Théorème 2.5].

Théorème 2.1. *Pour tout q puissance d'un nombre premier et m un entier non nul, alors*

il existe un corps fini contenant q^m. De plus, deux corps finis ayant le même nombre d'éléments sont isomorphes.

Dans ce qui suit, $\mathbb{F}_q$ est un corps fini à q éléments, tel que $q \geq 2$ est une puissance d'un nombre premier.

Soient a et b deux éléments de $\mathbb{F}_q$, un simple calcul montre que $(a+b)^q = a^q + b^q$. On peut généraliser aux puissances supérieures, on a $(a+b)^{q^l} = a^{q^l} + b^{q^l}$, $\forall l \in \mathbb{N}$.

Théorème 2.2. *Soit* P *un polynôme irréductible dans* $\mathbb{F}_q[X]$ *de degré* m. *Le corps* $\mathbb{F}_{q^m}$ *est isomorphe à l'anneau des polynômes* $\mathbb{F}_q[X]$ *modulo* P.

On peut construire un corps de taille q^m à partir du corps $\mathbb{F}_q$ et d'un polynôme irréductible de degré m. Chaque polynôme P de même degré donne une représentation différente du corps $\mathbb{F}_{q^m}$.

Théorème 2.3. *Le groupe multiplicatif* $\mathbb{F}_{q^m}^*$ *est cyclique.*

Le générateur d'un groupe cyclique s'appelle élément primitif. On a

$$\mathbb{F}_{q^m} = \left\{\alpha^i, i \in \left[1, q^m - 1\right]\right\} \cup \{0\}.$$

2.2 Les q-polynômes

Les q-polynômes, sont appelés parfois polynômes linéarisés et ont été étudiés par Öre dans [Ore33]. Ces polynômes sont de très grande importance et ont beaucoup de domaines d'applications dans la théorie comme dans la pratique, en particulier dans le domaine de la théorie des codes. L'utilité de ces polynômes vient du fait que la détermination du polynôme linéarisé est équivalente à trouver la base de ses racines. Nous nous servirons des q-polynômes pour la récupération de la base de l'erreur en utilisant les codes de Gabidulin et les codes LRPC.

Définition 2.1. Un q-polynôme P(X) de q-degré n dans $\mathbb{F}_{q^m}$ est un polynôme de la forme :

$$\mathrm{P}(\mathrm{X}) = p_0\mathrm{X} + p_1\mathrm{X}^q + \cdots + p_n\mathrm{X}^{q^n}, \tag{2.1}$$

où $p_0, p_1, \ldots, p_n$ sont des éléments de $\mathbb{F}_{q^m}$ et $p_n \neq 0$.

Nous pouvons tirer deux importantes propriétés de la définition précédente :

— $\mathrm{P}(a+b) = \mathrm{P}(a) + \mathrm{P}(b)$ pour tout $a, b \in \mathbb{F}_{q^m}$.

— $\mathrm{P}(\alpha a) = \alpha \mathrm{P}(a)$ pour tout $\alpha \in \mathbb{F}_q$ et $a \in \mathbb{F}_{q^m}$.

Ainsi, l'ensemble des q-polynômes est un sous-espace vectoriel de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$. En particulier, toute combinaison des racines d'un q-polynôme P est aussi une racine de P.

2.2.1 L'anneau des q-polynômes

La somme de deux q-polynômes est un q-polynôme. Par conséquent, l'ensemble des q-polynômes muni de la loi d'addition usuelle est un groupe abélien. Cependant, la multiplication de deux q-polynômes n'est pas forcement un q-polynôme, i.e. $\exists (i,j) \in \mathbb{N}^2$ tels que $\forall m \in \mathbb{N}$ on a $q^m \neq q^i + q^j$. Cela veut dire que l'ensemble de q-polynômes n'est pas stable par la multiplication usuelle. On définit donc une deuxième loi de composition interne pour les q-polynômes que l'on appellera le produit symbolique.

Le produit symbolique de deux q-polynômes est défini comme la composition des deux q-polynômes, i.e.

$$\mathrm{P}(\mathrm{X}) \otimes \mathrm{Q}(\mathrm{X}) \triangleq \mathrm{P}(\mathrm{Q}(\mathrm{X})).$$

On peut vérifier que l'ensemble de q-polynômes est stable par le produit symbolique. En effet,

$$\mathrm{P}(\mathrm{X}) \otimes \mathrm{Q}(\mathrm{X}) = \sum_{i=0}^{n_p} p_i \left(\sum_{j=0}^{n_q} q_j \mathrm{X}^{q^j} \right)^{q^i} = \sum_{i=0}^{n_p} \sum_{j=0}^{n_q} p_i q_j q^i \mathrm{X}^{q^{i+j}},$$

avec $\mathrm{P}(\mathrm{X}) = \sum_{i=0}^{n_p} p_i \mathrm{X}^{q^i}$ et $\mathrm{Q}(\mathrm{X}) = \sum_{j=0}^{n_q} q_j \mathrm{X}^{q^j}$.

Par ailleurs, le produit symbolique de deux q-polynômes est bien un q-polynôme. Ainsi, l'ensemble des q-polynômes à coefficients dans $\mathbb{F}_{q^m}$ muni de loi d'addition et de produit symbolique est un anneau non commutatif avec $x^{q^0} = x$, qui est l'élément neutre du produit symbolique. Le produit symbolique de deux polynômes $\mathrm{P}(\mathrm{X})$ et $\mathrm{Q}(\mathrm{X})$, respectivement, de q-degré n_p et n_q vérifie :

$$\mathrm{D}_1(\mathrm{X}) = \mathrm{P}(\mathrm{X}) \otimes \mathrm{Q}(\mathrm{X}) = \sum_{i=0}^{n_p} \sum_{j=0}^{n_q} p_i q_j^{q^i} \mathrm{X}^{q^{i+j}}$$

$$\mathrm{D}_2(\mathrm{X}) = \mathrm{P}(\mathrm{X}) \otimes \mathrm{Q}(\mathrm{X}) = \sum_{i=0}^{n_p} \sum_{j=0}^{n_q} p_i^{q^j} q_j \mathrm{X}^{q^{i+j}}.$$

En général, le produit symbolique n'est pas commutatif. Cependant, le q-degré de $\mathrm{D}_1(\mathrm{X})$ et $\mathrm{D}_2(\mathrm{X})$ est égal à $n_p + n_q$.

La propriété fondamentale des polynômes qui montre la relation entre l'espace vectoriel et les q-polynômes est introduite dans le théorème suivant.

Théorème 2.4. *Soit* V *un sous-espace vectoriel de* $\mathbb{F}_{q^m}$ *de dimension* n *sur* $\mathbb{F}_q$. *Le polynôme*

$$\mathrm{P}(\mathrm{X}) = \prod_{v \in \mathrm{V}} (\mathrm{X} - v)$$

est un q-polynôme. De plus, $\mathrm{P}(\mathrm{X})$ *est l'unique q-polynôme unitaire de q-degré* n *dont les racines sont tous dans* V.

Ce théorème est la solution clé pour le décodage des codes en métrique rang. Le fait de trouver le q-polynôme qui a comme racine des éléments de l'espace d'erreur est équivalent à trouver l'espace lui même.

2.2.2 La division euclidienne à droite

L'ensemble des q-polynômes est un anneau non commutatif. Cela veut dire qu'il faut définir une division à droite et une division euclidienne à gauche. Dans cette thèse, nous avons besoin de la division à droite des q-polynômes dans l'algorithme de décodage des codes en métrique rang que nous introduisons dans la suite de ce chapitre et dans le prochain chapitre. Par conséquent, nous définissons uniquement la division à droite des q-polynômes et quelques-unes de ses propriétés.

Définition 2.2. Soient $\mathrm{P}(\mathrm{X})$ et $\mathrm{S}(\mathrm{X})$ deux q-polynômes. $\mathrm{S}(\mathrm{X})$ est divisible à droite par $\mathrm{P}(\mathrm{X})$ s'il existe un q-polynôme tel que $\mathrm{S}(\mathrm{X}) = \mathrm{Q}(\mathrm{X}) \otimes \mathrm{P}(\mathrm{X})$.

Proposition 2.1 (Divisions euclidiennes à droite).
Soient $\mathrm{P}(\mathrm{X})$ *et* $\mathrm{S}(\mathrm{X})$ *deux q-polynômes. Il existe deux uniques* $\mathrm{Q}(\mathrm{X})$ *et* $\mathrm{R}(\mathrm{X})$ *q-polynômes tel que* $\mathrm{S}(\mathrm{X}) = \mathrm{Q}(\mathrm{X}) \otimes \mathrm{P}(\mathrm{X}) + \mathrm{R}(\mathrm{X})$, $deg_q(\mathrm{R}(\mathrm{X})) < deg_q(\mathrm{P}(\mathrm{X}))$.

Soient $\mathrm{P}(\mathrm{X})$ et $\mathrm{S}(\mathrm{X})$ deux q-polynômes. On note $\mathrm{RD}iv(\mathrm{S}(\mathrm{X}), \mathrm{P}(\mathrm{X}))$ la fonction qui renvoie le reste de la division euclidienne à droite de $\mathrm{S}(\mathrm{X})$ sur $\mathrm{P}(\mathrm{X})$.

Algorithme 1 La division à droite de deux q-polynômes [Ore33]

Entrée:
$\mathrm{S}(\mathrm{X}), \mathrm{P}(\mathrm{X})$, ▷ Deux q-polynômes
Sortie:
$\mathrm{R}(\mathrm{X})$, ▷ Le reste de la division à droite de $\mathrm{S}(\mathrm{X})$ sur $\mathrm{P}(\mathrm{X})$

1: $d_{\mathrm{S}} \leftarrow deg_q(\mathrm{S}(\mathrm{X}))$
2: $d_{\mathrm{P}} \leftarrow deg_q(\mathrm{P}(\mathrm{X}))$
3: **tant que** $d_{\mathrm{S}} \geq d_{\mathrm{P}}$ **faire**
4: $\quad \mathrm{S}(\mathrm{X}) \leftarrow \mathrm{S}(\mathrm{X}) - \frac{s_{d_{\mathrm{S}}}}{p_{d_{\mathrm{P}}}^{q^{d_{\mathrm{S}}-d_{\mathrm{P}}}}} \mathrm{X}^{q^{d_{\mathrm{S}}-d_{\mathrm{P}}}} \otimes \mathrm{P}(\mathrm{X})$
5: $\quad d_{\mathrm{S}} \leftarrow deg_q(\mathrm{S}(\mathrm{X}))$
6: **fin tant que**
7: $\mathrm{R}(\mathrm{X}) \leftarrow \mathrm{S}(\mathrm{X})$

On peut utiliser cet algorithme pour calculer le reste de la division à droite de deux q-polynômes. Le nombre maximal d'itérations de l'algorithme 1 est inférieur ou égal à $deg_q(\mathrm{S}(\mathrm{X})) - deg_q(\mathrm{P}(\mathrm{X})) + 1$.

Proposition 2.2. *Soient* $\mathrm{S}(\mathrm{X})$ *et* $\mathrm{P}(\mathrm{X})$ *deux q-polynômes à coefficient dans* $\mathbb{F}_{q^m}$. $\mathrm{P}(\mathrm{X})$ *est un diviseur à droite de* $\mathrm{S}(\mathrm{X})$ *si et seulement si* $\mathrm{P}(\mathrm{X})$ *est un diviseur classique de* $\mathrm{S}(\mathrm{X})$.

Démonstration. Supposons que $\mathrm{P}(\mathrm{X})$ est un diviseur à droite de $\mathrm{S}(\mathrm{X})$. alors, il existe un unique polynôme $\mathrm{Q}(\mathrm{X})$ tel que $\mathrm{S}(\mathrm{X}) = \mathrm{Q}(\mathrm{X}) \otimes \mathrm{P}(\mathrm{X})$. Or, $\mathrm{Q}(\mathrm{X}) \otimes \mathrm{P}(\mathrm{X}) = \sum_{i=0}^{n_q} q_j (\mathrm{P}(\mathrm{X}))^{q^j}$ est

divisible par P dans le sens classique. D'où, $P(X)$ est un diviseur classique de $S(X)$.
Inversement, $S(X)$ et $P(X)$ sont deux q-polynômes, alors, il existe deux q-polynômes $Q(X)$ et $R(X)$ tels que $S(X) = Q(X) \otimes P(X) + R(X)$ et $deg_q(R(X)) < deg_q(P(X))$. Le q-polynôme $Q(X) \otimes P(X) + R(X)$ peut aussi s'écrire sous la forme $\widetilde{Q}(X) \times P(X)$ où $\widetilde{Q}$ n'est pas forcement un q-polynôme. Par conséquent, $S(X) = \widetilde{Q}(X) \times P(X) + R(X)$ et $deg_q(R(X)) < deg_q(P(X))$. Donc, $deg(R(X)) < deg(P(X))$. Puisque, $P(X)$ est un diviseur classique de $S(X)$ de reste 0, alors, on a $R(X) = 0$ par unicité. □

2.2.3 Le plus grand diviseur commun à droite

Dans la partie précédente, nous avons défini la division euclidienne à droite. Dans cette partie, nous introduisons la notion de *plus grand diviseur commun à droite* ou RGCD. Par la suite, nous montrons la relation entre l'intersection de deux sous-espaces vectoriels et le plus grand diviseur commun à droite de deux q-polynômes.

Définition 2.3. Soient $P_1(X)$ et $P_2(X)$ deux q-polynômes à coefficient dans $\mathbb{F}_{q^m}$. Alors, il existe un unique q-polynôme unitaire de q-degré maximal qui divise $P_1(X)$ et $P_2(X)$. On appelle ce q-polynôme le plus grand diviseur commun à droite de $P_1(X)$ et $P_2(X)$ et on le note $RGCD(P_1, P_2)$.

Notons que le plus grand diviseur commun de deux q-polynômes au sens classique est égal au plus grand diviseur commun à droite. La prochaine proposition illustre le lien entre l'intersection de deux espace vectoriels et les q-polynômes.

Proposition 2.3. *Soient* E *et* F *deux sous-espaces vectoriels de* $\mathbb{F}_{q^m}$ *et soient* $P_E(X)$ *et* $P_F(X)$*, respectivement, leurs q-polynômes associés. Le q-polynôme associé de l'intersection de* E *et* F *est*

$$P_{E\cap F}(X) = RGCD\left(P_E(X), P_F(X)\right).$$

Démonstration. En utilisant la proposition 2.2, on peut facilement prouver que $P_{E\cap F}(X)$ est un diviseur symbolique à droite de $RGCD\left(P_E(X), P_F(X)\right)$.
Inversement, si $X - a$ divise $RGCD\left(P_E(X), P_F(X)\right)$, alors, $X - a$ divise $P_E(X)$ et $P_F(X)$. Cela veut dire que a appartient à E et F. Donc, $RGCD(P_E(X), P_F(X))$ est un diviseur classique de $P_{E\cap F}(X)$. Selon la proposition 2.2, $RGCD(P_E(X), P_F(X))$ est un diviseur symbolique à droite de $P_{E\cap F}(X)$. □

Le plus grand diviseur commun à droite des deux polynômes peut être calculé en utilisant l'algorithme d'Euclide étendu décrit dans [LNC83].

Supposons qu'on connaisse le q-degré de $RGCD(P_1, P_2)$. Le dernier reste non nul de la division de P_1 et P_2 est $RGCD(P_1, P_2)$ et son q-degré est égal à r.

Algorithme 2 Algorithme d'Euclide étendu

Entrée:
- $P_1(X)$, $P_2(X)$ ▷ Deux q-polynômes avec $deg_q(P_1(X) \geq deg_q(P_2(X))$
- r ▷ Le q-degré de $RGCD(P_1(X), P_2(X))$

Sortie:
- $R(X)$ ▷ Le plus grand diviseur commun à droite $P_1(X)$ et $P_2(X)$

```
R_1(X) ← P_1(X)
R_2(X) ← P_2(X)
tant que deg_q(R_2(X)) ≠ r faire
    R_3(X) ← RDiv(R_1(X), R_2(X))        ▷ Calculé en utilisant l'algorithme 1
    R_1(X) ← R_2(X)
    R_2(X) ← R_3(X)
fin tant que
R(X) ← R_1(X)
```

2.3 Présentation des codes et leurs propriétés

On se place dorénavant dans le corps finis $\mathbb{F}_{q^m}$. Pour définir un code, comme étant un espace métrique, nous avons besoin de définir une distance sur cette espace pour mesurer la distance entre ses éléments. Cette distance doit vérifier les conditions suivantes.

Définition 2.4. L'application $dist$ définie par

$$\begin{aligned} dist: \quad & \mathbb{F}_{q^m}^{\mathrm{N}} \times \mathbb{F}_{q^m}^{\mathrm{N}} \to \mathbb{N} \\ & (x, y) \mapsto dist(x, y), \end{aligned}$$

est une distance sur $\mathbb{F}_{q^m}^{\mathrm{N}}$ si elle vérifie les trois propriétés suivantes :

— la symétrie : $dist(x, y) = dist(y, x)$, $\forall (x, y) \in \mathbb{F}_{q^m}^{\mathrm{N}} \times \mathbb{F}_{q^m}^{\mathrm{N}}$.

— la séparation : $dist(x, y) = dist(y, x) = 0 \Rightarrow x = y$.

— l'inégalité triangulaire : $dist(x, y) \leq dist(x, z) + dist(z, y)$, $\forall (x, y, z) \in \left(\mathbb{F}_{q^m}^{\mathrm{N}}\right)^3$.

Un code $\mathcal{C}$ sur $\mathbb{F}_{q^m}$ de longueur N et de cardinal M est un sous ensemble de l'espace de mot $\mathbb{F}_{q^m}^{\mathrm{N}}$. Notons $\mathrm{R} = \frac{log_{q^m}(\mathrm{M})}{\mathrm{N}}$ *le rendement* du code $\mathcal{C}$. De façon globale, nous pouvons associer à chaque code sa distance minimale que l'on définit comme suit.

Définition 2.5 (Distance minimale).
Soit $\mathcal{C}$ un code de longueur N, la distance minimale d_m est définie par

$$d_m = \min_{c_1 \neq c_2 \in \mathcal{C}} (dist(c_1, c_2)).$$

On définit la capacité de correction comme : $t = \left\lfloor \frac{d_m - 1}{2} \right\rfloor$.

La proposition suivante donne une condition suffisante pour trouver le mot de code le plus proche du mot reçu.

Proposition 2.4. *Soit $\mathcal{C}$ un code sur $\mathbb{F}_{q^m}$ de longueur* N*, de cardinal* M *et de distance minimale d_m. Soit y un mot de longueur* N *de distance à un mot de code c inférieure à t la capacité de correction de $\mathcal{C}$. Alors c est l'unique mot de code de $\mathcal{C}$ de distance à y inférieure à t.*

2.3.1 Codes en bloc linéaires

Les codes les plus utilisés dans la pratique sont les codes linéaires, c-à-d les codes définis comme étant des sous-espaces vectoriels de $\mathbb{F}_{q^m}^{\mathrm{N}}$. Ces codes sont caractérisés par leur structure qui facilite leur représentation sans avoir besoin de définir tout l'ensemble $\mathcal{C}$. Dans cette thèse, nous ne nous intéressons qu'aux codes en bloc linéaires sans préciser d'avantage. Dans la pratique, un code en bloc de longueur N est un code dans lequel chaque bloc de longueur N est décodé indépendamment des autres blocs. Nous supposons, sauf mention contraire, qu'un code est un code en bloc linéaire.

Définition 2.6 (Code linéaire).
Un code linéaire $\mathcal{C}$ de longueur N, de distance minimale d_m et de dimension k sur $\mathbb{F}_{q^m}$ est un sous espace vectoriel de $\mathbb{F}_{q^m}^{\mathrm{N}}$. Un tel code est dit de paramètres $[\mathrm{N}, k, d]$.

Un code linéaire $\mathcal{C}$ peut être défini par une matrice taille $k \times \mathrm{N}$ appelée *matrice génératrice*, elle sera notée $\mathbf{G}$. Pour que le code soit de dimension k, il faut que les lignes de sa matrice $\mathbf{G}$ forment une base de $\mathcal{C}$ sur $\mathbb{F}_{q^m}$.

Dans la pratique, on utilise la matrice génératrice pour calculer un mot de code à partir du mot d'information. La matrice $\mathbf{G}$ permet de définir un code linéaire de façon unique. Cependant, un code $\mathcal{C}$ possède plusieurs matrices génératrices. On choisit donc une matrice génératrice, de la forme la plus simple possible, pour représenter le code. Une matrice génératrice est sous forme *systématique* si elle est composée d'un bloc identité et d'une matrice quelconque.

Définition 2.7 (Code dual).
Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d]$. Le code dual de $\mathcal{C}$, noté $\mathcal{C}^{\perp}$, est définie par :

$$\mathcal{C}^{\perp} = \left\{ c' \in \mathbb{F}_{q^m}^{\mathrm{N}} \mid \langle c, c' \rangle = 0, \forall c \in \mathcal{C} \right\},$$

où $\langle c, c' \rangle = \sum_{i=1}^{\mathrm{N}} c_i c'_i$ est le produit scalaire des vecteurs c et c'.

Un code dual de $\mathcal{C}$ est un code linéaire de paramètres $\left[\mathrm{N}, \mathrm{N}-k, d^{\perp}\right]$. Sa distance minimale $d^{\perp}$ ne dépend pas, en général, des paramètres du code $\mathcal{C}$. Sa matrice génératrice $\mathbf{H}$, *appelée matrice de parité* du code $\mathcal{C}$, est formée par la base du noyau de $\mathbf{G}$. Ainsi on peut définir le code $\mathcal{C}$ de la manière suivante :

$$\mathcal{C} = \left\{ x\mathbf{G}, x \in \mathbb{F}_{q^m}^{k} \right\} = \left\{ c \in \mathbb{F}_{q^m}^{\mathrm{N}}, \mathbf{H}c^{\perp} = 0 \right\}.$$

Définition 2.8 (Syndrome).
Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d]$ et de matrice de parité $\mathbf{H}$. Soit y un mot de $\mathbb{F}_{q^m}^{\mathrm{N}}$, le vecteur $s = y\mathbf{H}^{\perp}$ est appelé *le syndrome de y*.

Pour avoir une idée sur la distance minimale dans le cas des codes linéaires, nous disposons de la proposition suivante qui donne les conditions pour que la distance minimale soit égale à une valeur d_m.

Proposition 2.5. *Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d_m]$ et de matrice de parité $\mathbf{H}$. Alors*

- $\forall e \in \mathbb{F}_{q^m}^{\mathrm{N}}$ *avec* $d(0, e) \leq d_m - 1$, *on a* $e\mathbf{H}^{\perp} \neq 0$.
- $\exists c \in \mathcal{C}$ *tel que* $d(0, c) = d_m$.

2.3.2 Le principe de décodage

Lors de la transmission du mot de code, certains symboles sont erronés lors de leurs passage par le canal. A la sortie du canal, le mot reçu y est ainsi erroné. En général, y n'est pas dans $\mathcal{C}$, mais dans l'espace $\mathbb{F}_{q^m}^{\mathrm{N}}$. Le principe de décodage consiste à récupérer le mot transmis le plus proche du mot reçu. Nous utilisons, dans cette thèse, le terme *"nombre d'erreurs"* pour designer la distance entre le mot de code et le mot reçu. Nous appelons *"le vecteur erreur"* la différence entre le mot envoyé et le mot reçu.

Le but de la théorie des codes est de trouver un algorithme de décodage qui soit efficace pour la correction des erreurs. Un code est dit *efficace* s'il possède un algorithme de décodage qui peut corriger un nombre maximal d'erreurs. Nous allons voir que l'efficacité d'un code dépend du canal utilisé et de ses paramètres.

Pour comparer les performances de deux codes, il faut utiliser le même canal, la même longueur de code et le même rendement. Par ailleurs, il faut comparer la complexité de leurs algorithmes de codage et décodage. Parfois, il faut comparer aussi la complexité circuit des deux codes pour juger l'efficacité d'un code.

Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d]$ et de matrice de parité $\mathbf{H}$. Supposons qu'on reçoit un mot y de $\mathbb{F}_{q^m}^{\mathrm{N}}$.

Décodage par distance minimale

L'algorithme décodage par distance minimale compare y aux mots de code de $\mathcal{C}$ et renvoie le mot de code le plus proche de y en terme de distance. Dans le cas d'un canal discret sans mémoire, où la probabilité d'avoir une erreur de poids faible est importante, le décodage par distance minimale et *le décodage à maximum de vraisemblance* sont équivalents. Après avoir appliqué le décodage par distance minimale, l'algorithme renvoie la valeur suivante

$$c' = arg \min_{c \in \mathcal{C}} d\left(y, c\right).$$

S'il existe plusieurs mots de code de même distance à y, on choisit d'une façon aléatoire un mot de code parmi les mots retrouvés. Comme nous avons vu dans la proposition 2.4, dans le cas où la distance à un mot de code est inférieure à la capacité de correction de $\mathcal{C}$, le décodage par distance minimale garantit un résultat unique.

La Figure 1 illustre un exemple de fonctionnement de l'algorithme de décodage par distance minimale. On suppose que le mot reçu y est à distance $d(y, c_1)$ de c_1 et $d(y, c_2)$ de c_2. On remarque que $d(y, c_2)$ est plus petite que $d(y, c_1)$, dans ce cas l'algorithme de décodage renvoie c_2.

Décodage par syndrome

Le syndrome d'un mot reçu ne dépend pas du mot de code, il dépend uniquement de l'erreur. Pour cette raison, la majorité des codes en bloc utilisent cette méthode de décodage. Le principe du décodage par syndrome est de diviser l'ensemble des mots

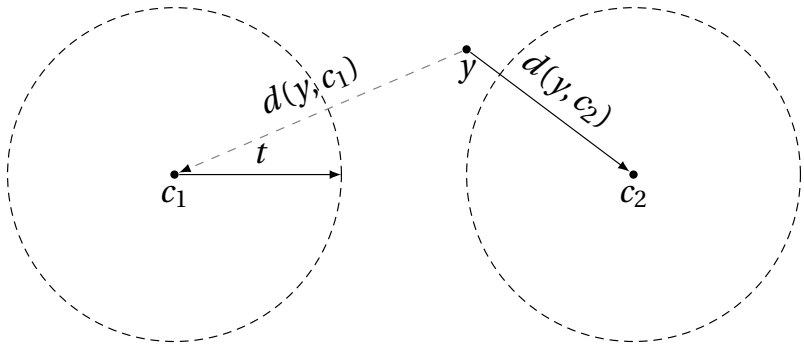


FIGURE 1 – Exemple de fonctionnement de l'algorithme de décodage par distance minimale.

de $\mathbb{F}_{q^m}^{\mathrm{N}}$ en $q^{\mathrm{N}-k}$ classes, chaque classe correspond à une valeur de syndrome. On choisit pour chaque classe un représentant de petit poids. Par exemple, pour le syndrome 0 la classe qui correspond est $\mathcal{C}$, le représentant de cette classe est le vecteur nul. Ensuite, on calcule le syndrome du mot reçu $s = y\mathbf{H}$. On note e le représentant de cette classe. Finalement, le mot émis serait $c' = y - e$, où e est le représentant de la classe de s.

Dans ce cas de décodage, on est sûr que le résultat à la sortie de l'algorithme est unique. Pour avoir une idée sur la performance de décodage par syndrome, nous avons la proposition suivante.

Proposition 2.6. *Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d_m]$ et de matrice de parité $\mathbf{H}$. Supposons que l'on reçoit un mot $y = c + e$ de $\mathbb{F}_{q^m}^{\mathrm{N}}$. Si le poids de l'erreur e est inférieur à la capacité de correction du code $\mathcal{C}$ alors on peut récupérer c.*

Démonstration. Soient $e_1, e_2 \in \mathbb{F}_{q^m}^{\mathrm{N}}$ et $e_1 \neq e_2$ tel que :

$$e_1 \mathrm{H}^{\perp} = e_2 \mathrm{H}^{\perp} = s.$$

Supposons que les poids de e_1 et e_2 sont inférieurs à $\left\lfloor \frac{d_m - 1}{2} \right\rfloor$. Alors, $(e_1 - e_2)\,\mathrm{H}^{\perp} = 0$. Cela implique que $e_1 - e_2$ est un mot de code. Donc,

$$\begin{aligned} d\,(0, e_1 - e_2) &\leq d\,(e_1, 0) + d\,(0, e_2) \\ &\leq \left\lfloor \frac{d_m - 1}{2} \right\rfloor + \left\lfloor \frac{d_m - 1}{2} \right\rfloor \\ &\leq d_m - 1, \end{aligned}$$

ce qui est absurde. Alors, dans chaque classe il existe au plus un représentant de poids inférieur à $\left\lfloor \frac{d_m - 1}{2} \right\rfloor$. □

On considère l'exemple de décodage par syndrome dans la Figure 2. On remarque que la distance entre le mot y et le mot de code c_2, notée w est inférieure à la capacité de correction t. Donc, l'algorithme de décodage renvoie la valeur c_2.

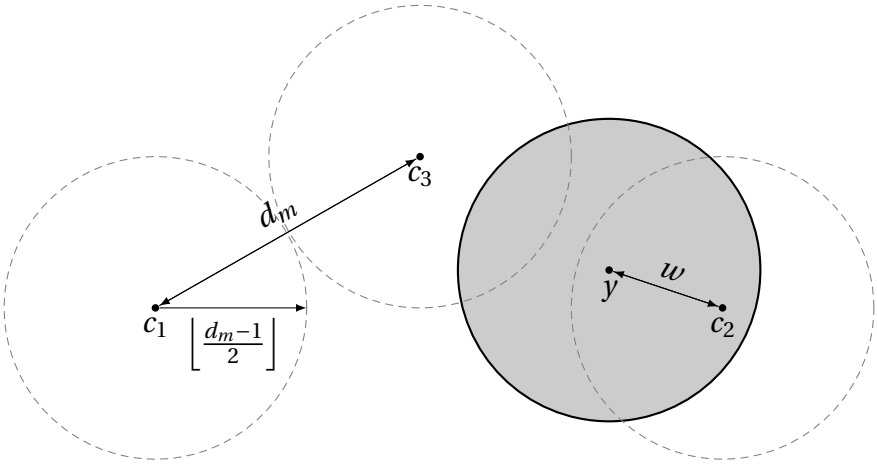


FIGURE 2 – Exemple de décodage par syndrome dans le cas où le poids de l'erreur $w \leq t$.

2.4 Rappel sur la métrique rang

Dans cette partie, nous introduisons la métrique rang ainsi que ses principales propriétés. Ensuite, nous définissons deux bornes dans cette métrique, qui sont la borne de Singleton et la borne de Gilbert-Varshamov.

2.4.1 Définitions et propriétés

On fixe, pour la suite de ce chapitre, une base quelconque de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$ notée $\mathcal{B} = \{\beta_1, \beta_2, \ldots, \beta_m\}$. Chaque élément de $\mathbb{F}_{q^m}$ s'écrit sous la forme

$$x = \sum_{i=1}^{m} x_i \beta_i$$

où $x_{i=1:m}$ sont des éléments du corps de base $\mathbb{F}_q$. Cela veut dire que $\mathbb{F}_{q^m}$ est en bijection avec l'espace vectoriel $\mathbb{F}_q^m$. On peut donc définir une application linéaire bijective donnée par

$$\begin{aligned} ext_{\mathcal{B}} : \quad \mathbb{F}_{q^m} &\to \mathbb{F}_q^{m \times \mathrm{N}} \\ x &\mapsto \mathbf{X}, \end{aligned}$$

où $\mathbf{X}$ est une matrice de $\mathbb{F}_q^{m \times \mathrm{N}}$ qui vérifie

$$x_j = \sum_{i=1}^{m} \mathrm{X}_{i,j} \beta_i, \forall j \in [\![1, \mathrm{N}]\!].$$

La matrice associée à x est $ext_{\mathcal{B}}(x)$ ou bien $\mathbf{X}$ s'il n'y a pas d'ambiguïté.

Définition 2.9. Soit $x = (x_1, x_2, \ldots, x_{\mathrm{N}})$ un vecteur de $\mathbb{F}_{q^m}^{\mathrm{N}}$. On appelle le *"rang"* de x sur $\mathbb{F}_q$ le rang de sa matrice associée X.

Le rang de x sera noté par la suite $rang(x \mid \mathbb{F}_q)$ ou plus simplement $rang(x)$ si il n'y a pas de confusion. Le $rang(x \mid \mathbb{F}_q)$ ne dépend pas du choix de la base $\mathcal{B}$, autrement dit, pour tout $\mathcal{B}$ et $\mathcal{B}'$ bases de $\mathbb{F}_{q^m}$, $rang(ext_{\mathcal{B}}(x)) = rang(ext_{\mathcal{B}'(x)})$.

Nous avons besoin d'introduire une métrique, comme dans le cas des codes en métrique de Hamming, pour définir un code correcteur d'erreur en métrique rang. Nous vérifions que le rang d'un vecteur est bien une métrique.

Définition 2.10. L'application d_r définie par

$$\begin{aligned} d_r : \quad & \mathbb{F}_{q^m}^{\mathrm{N}} \times \mathbb{F}_{q^m}^{\mathrm{N}} \to \mathbb{N} \\ & (x, y) \mapsto d_r(x, y) = rang(x - y), \end{aligned}$$

est une distance sur $\mathbb{F}_{q^m}^{\mathrm{N}}$.

Dans le cas des codes linéaires en métrique rang, on dispose d'un théorème donnant une condition nécessaire et suffisante sur leurs distances rang minimales [Gab85].

Proposition 2.7. *Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_{q^m}$ de paramètres $[\mathrm{N}, k, d_m]$ et de matrice de parité $\mathbf{H}$. Alors*

— $\forall \mathbf{Y} \in \mathbb{F}_{q^m}^{(d-1)\times \mathrm{N}}$ *tel que* $\mathbf{Y}$ *est de rang plein, on a* $\left(\mathbf{Y}\mathbf{H}^{\perp}\right) = d_m - 1$.

— $\exists \mathbf{Y}_0 \in \mathbb{F}_{q^m}^{(d)\times \mathrm{N}}$ *tel que* $\mathbf{Y}_0$ *est de rang plein, on a* $\left(\mathbf{Y}_0\mathbf{H}^{\perp}\right) < d_m$.

2.4.2 Bornes sur les codes en métrique rang

Le nombre de sous espaces vectoriels de $\mathbb{F}_{q^m}$ de dimension t sur $\mathbb{F}_q$ est calculé par :

$$\begin{bmatrix} m \\ t \end{bmatrix}_q \triangleq \prod_{i=0}^{t-1} \frac{q^m - q^i}{q^t - q^i}, \tag{2.2}$$

la quantité $\begin{bmatrix} m \\ t \end{bmatrix}_q$ est appelée *le coefficient Gaussien.*

On définit le nombre de sphère en métrique rang de rayon t autour de 0 comme étant le cardinal de l'ensemble des vecteurs de $\mathbb{F}_{q^m}^{\mathrm{N}}$ de rang égal à t. Ce dernier est noté $\mathrm{S}(m, \mathrm{N}, q, t)$. Alors, on a

$$\mathrm{S}(m, \mathrm{N}, q, t) = \begin{bmatrix} m \\ t \end{bmatrix}_q \prod_{i=0}^{t-1} \left(q^{\mathrm{N}} - q^i\right). \tag{2.3}$$

On peut calculer le nombre de matrices de rang t dans $\mathbb{F}_q^{m\times \mathrm{N}}$ en utilisant l'équation (2.3). Supposons que $m \geq \mathrm{N}$, le nombre de matrices de rang plein dans $\mathbb{F}_q^{m\times \mathrm{N}}$ est :

$$\mathrm{S}(m, \mathrm{N}, q, \mathrm{N}) = q^{m\mathrm{N}} \prod_{i=0}^{\mathrm{N}-1} \left(1 - q^{i-m}\right). \tag{2.4}$$

A partir de l'équation (2.3), on déduit le nombre de vecteurs de $\mathbb{F}_{q^m}^{\mathrm{N}}$ de rang inférieur ou égal à t

$$\mathrm{B}(m, \mathrm{N}, q, t) = \sum_{j=0}^{t} \mathrm{S}(m, \mathrm{N}, q, j) = \sum_{j=0}^{t} \begin{bmatrix} m \\ j \end{bmatrix}_q \prod_{i=0}^{j-1} \left(q^{\mathrm{N}} - q^i\right). \tag{2.5}$$

Comme dans le cas de métrique de Hamming, nous pouvons définir *la borne de Singleton* et *la borne de Gilbert-Varshamov* en métrique rang [Del78] ; [GY06].

Borne de Singleton

Soit $\mathcal{C}$ un code de longueur N et de cardinal M et de distance rang minimale d_m. Alors, on a

$$\mathrm{M} \leq q^{(\min(m,\mathrm{N})-d+1)\max(m,\mathrm{N})}. \tag{2.6}$$

Dans le cas d'égalité, le code $\mathcal{C}$ est appelé *code Maximum Distance Rang* (MDR) ou *code optimal*. Pour les codes linéaires, la distance minimale vérifie

— $d \leq \left\lfloor \frac{m(\mathrm{N}-k)}{\mathrm{N}} \right\rfloor + 1$, si $m \leq \mathrm{N}$.

— $d \leq \mathrm{N} - k + 1$, si $m > \mathrm{N}$.

Borne de Gilbert-Varshamov

Soit $\mathcal{C}$ un code de longueur N et de cardinal M et de distance rang minimale d_m. Le code $\mathcal{C}$ atteint la borne de Gilbert-Varshamov (RGV) si :

$$(\mathrm{M}-1) \times \mathrm{B}(m, \mathrm{N}, q, d_m - 1) < q^{m\mathrm{N}} \leq \mathrm{M} \times \mathrm{B}(m, \mathrm{N}, q, d_m - 1).$$

Dans le cas des codes linéaires, la borne de Gilbert-Varshamov correspond à un entier r à partir duquel, pour tout syndrome s, il existe en moyenne un mot e de rang égal à r tel que $e\mathbf{H}^{\perp} = s$. Pour avoir une idée sur le comportement de borne la RGV quand m et N tend vers l'infini, on a l'approximation suivante [Loi14] :

$$\mathrm{RGV}\,(m, k, \mathrm{N}) \sim \frac{m + \mathrm{N} - \sqrt{(m-\mathrm{N})^2 + 4km}}{2}. \tag{2.7}$$

2.5 Les codes Gabidulin

Les codes Gabidulin ont été introduits en 1985 dans l'article [Gab85]. Gabidulin a présenté une manière de construire ces codes qui sont l'équivalent des codes Reed-Solomon pour la métrique de Hamming. Il parvient notamment à décrire un algorithme de décodage en temps polynomial jusqu'à la borne de Singleton.

Dans cette partie, nous définissons les codes optimaux. Puis, nous donnons un exemple de code qui est le code Gabidulin ainsi que ses principales propriétés. Ensuite, nous présentons la distance minimale des codes Gabidulin, leur matrice génératrice, leur matrice de parité et leur algorithme de décodage qui décode en temps polynomial.

2.5.1 Définition et propriétés

Soit P un q-polynôme et $x \in \mathbb{F}_{q^m}^{\mathrm{N}}$. On notera $\mathrm{P}(x) = (\mathrm{P}(x_1), ..., \mathrm{P}(x_n))$. Nous supposons dans toute la section que $m \geq \mathrm{N}$.

Définition 2.11. Soit $g = (g_1, g_2, \ldots, g_{\mathrm{N}}) \in \mathbb{F}_{q^m}$ une famille libre sur $\mathbb{F}_q$. Le code Gabidulin de paramètres $[\mathrm{N}, k]$ et de support g est l'ensemble des mots obtenus par évaluation d'un q-polynôme de degré au plus $k-1$ sur g :

$$\mathrm{G}ab(g,k,\mathrm{N}) = \left\{ \mathrm{P}(g) = \big(\mathrm{P}(g_1),...,\mathrm{P}(g_\mathrm{N})\big) \mid \quad \mathrm{P}(z) = \sum_{i=0}^{k-1} p_i z^{q^i} \right\}.$$

La relation entre les codes de Gabidulin et les codes Reed-Solomon généralisés (GRS) tient aux propriétés communes aux algèbres classiques de polynômes et aux algèbres de polynômes linéarisés.

Proposition 2.8 (Code optimal).
Le code gabidulin $\mathrm{G}ab(g,k,\mathrm{N})$ *est un code linéaire optimal en métrique rang. Il est capable de corriger jusqu'à* $\left\lfloor \frac{\mathrm{N}-k}{2} \right\rfloor$ *erreurs.*

La preuve de cette proposition est basée sur le fait que les N-uplets $(g_1, g_2, \ldots, g_\mathrm{N})$ sont linéairement indépendants. C'est pour cela que nous avons considéré le cas où la dimension du corps $\mathbb{F}_{q^m}$ est supérieure à N.

En se basant sur la définition 2.11, on peut définir la matrice génératrice d'un code Gabidulin. La matrice génératrice G du code $\mathrm{G}ab(g,k,\mathrm{N})$ est

$$\mathrm{G} = \begin{pmatrix} g_1 & g_2 & \cdots & g_\mathrm{N} \\ g_1^q & g_2^q & \cdots & g_\mathrm{N}^q \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_\mathrm{N}^{q^{k-1}} \end{pmatrix}. \tag{2.8}$$

Proposition 2.9. *La matrice génératrice* G *sous la forme systématique du code* $\mathrm{G}ab(g,k,\mathrm{N})$ *est*

$$\mathrm{G} = \left(\begin{array}{cccc|ccc} 1 & 0 & \cdots & 0 & \mathrm{P}_1(g_{k+1}) & \cdots & \mathrm{P}_1(g_\mathrm{N}) \\ 0 & 1 & \cdots & 0 & \mathrm{P}_2(g_{k+1}) & \cdots & \mathrm{P}_2(g_\mathrm{N}) \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \mathrm{P}_k(g_{k+1}) & \cdots & \mathrm{P}_k(g_\mathrm{N}) \end{array}\right)$$

où, pour $i = 1,\ldots,k$, P_i *est l'unique q-polynôme de q-degré k vérifiant, pour tout* $j \leq k$ *:* $\mathrm{P}_i(g_j) = \delta_{i,j}$.

La matrice de parité de $\mathrm{G}ab(g,k,\mathrm{N})$ est

$$\mathrm{H} = \begin{pmatrix} h_1 & h_2 & \cdots & h_\mathrm{N} \\ h_1^q & h_2^q & \cdots & h_\mathrm{N}^q \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{q^{d-1}} & h_2^{q^{d-1}} & \cdots & h_\mathrm{N}^{q^{d-1}} \end{pmatrix},$$

où, $h = (h_1, h_2, \ldots, h_\mathrm{N}) \in \mathbb{F}_{q^m}^\mathrm{N}$ est une solution non nulle des $\mathrm{N}-1$ équations à N inconnu

$$\sum_{i=1}^{\mathrm{N}} h_i^{q^{\mathrm{N}-k}} g_i^{q^j} = 0, \text{ pour } j = 1,2,\ldots,\mathrm{N}-1.$$

Ainsi, le code dual de $\mathrm{G}ab(g,k,\mathrm{N})$ est aussi un code Gabidulin de paramètres $[\mathrm{N}, \mathrm{N}-k]$ et de support h.

Théorème 2.5. *Le groupe des isométries semi-linéaires de* $\mathrm{G}ab(g,k,\mathrm{N})$ *est réduit au groupe des multiplications scalaires dans* $\mathbb{F}_{q^m}^*$. *En particulier, deux codes* $\mathrm{G}ab(g,k,\mathrm{N})$ *et* $\mathrm{G}ab(g',k,\mathrm{N})$ *sont égaux si et seulement s'il existe* $\alpha \in \mathbb{F}_{q^m}^*$ *tel que* $g = \alpha g'$.

2.5.2 Décodage des codes Gabidulin

Reconstruction de polynômes linéaires

Le problème de reconstruction de polynômes linéaires a été énoncé par Pierre Loidreau dans son article [Loi04]. Le décodage des codes de Gabidulin est un cas particulier de ce sujet. Tout ce qui suit est une réécriture de manière plus simple et plus compréhensible des travaux Pierre Loidreau.

Définition 2.12. (RqP(y, g, k, t)).
RqP est une fonction qui détermine tous les couples (V,P) de polynômes non nuls tels que $deg_q(\mathrm{V}) \leq t$, $deg_q(\mathrm{P}) \leq k-1$, et vérifiant $\mathrm{V}(y_i) = \mathrm{V} \circ \mathrm{P}(g_i)$, $i = 1,\ldots,n$.

Théorème 2.6. *Les polynômes linéaires* P*, pour lesquels il existe* V *tel que* (V,P) *est une solution de* $\mathrm{R}q\mathrm{P}(y,g,k,t)$*, sont exactement ceux associés à un mot de* $\mathrm{G}ab(g,k,n)$ *dont la distance à* y *est plus petite que* t*.*

Démonstration.

Supposons que (V,P) est solution de $\mathrm{R}q\mathrm{P}(y,g,k,t)$. Alors, on a $\mathrm{V}(y_i) = \mathrm{V} \circ \mathrm{P}(g_i)$, $i = 1,\ldots,n$, c-à-d $\mathrm{V}(y_i - \mathrm{P}(g_i)) = 0$. On a $1 \leq deg_q(\mathrm{V}) \leq t$. En posant $e_i = y_i - \mathrm{P}(g_i)$, $\mathrm{V}ect(e_1,\ldots,e_n)$ est un espace vectoriel de dimension au plus t, donc $rg(e) \leq t$. Ainsi, $\mathrm{P}(g) = y - e$ est un mot de $\mathrm{G}ab(g,k,n)$ dont la distance à y est plus petite que t.

Réciproquement, si $y = \mathrm{P}(g) + e$, et $rg(e) \leq t$, alors on peut, d'après la proposition 2.4, construire un q-polynôme V non nul tel que $deg_p(\mathrm{V}) \leq t$ et $\mathrm{V}(e) = 0$. Alors, pour $i \leq n$, $\mathrm{V}(yi) = \mathrm{V}(\mathrm{P}(g_i)) + \mathrm{V}(e_i) = \mathrm{V} \circ \mathrm{P}(g_i)$, donc (V,P) est une solution de $\mathrm{R}q\mathrm{P}(y,g,k,t)$. □

Conclusion : Résoudre $\mathrm{R}q\mathrm{P}(y,g,k,t)$ conduit à résoudre le système $\mathrm{V}(y_i) = \mathrm{V} \circ \mathrm{P}(g_i), \mathrm{V} \neq 0, i = 1,\ldots,n$.

On peut linéariser ce dernier, on obtient donc le système suivant :

$$\mathrm{V}(y_i) = \mathrm{N}(g_i), i = 1,\ldots,n, \mathrm{V} \neq 0$$

dont les inconnues sont $\mathrm{V}(\mathrm{X}) = \sum_{i=0}^{t} v_i \mathrm{X}^{[i]}$ et $\mathrm{N}(\mathrm{X}) = \sum_{i=0}^{k+t-1} n_i \mathrm{X}^{[i]}$.
Ce système est linéaire à n équations et $k+2t+1$ inconnues. Alors, le système précédant se traduit de façon matricielle :

$$\mathrm{S} \times \begin{pmatrix} \mathcal{N} \\ \mathcal{V} \end{pmatrix} = \begin{pmatrix} g_1 & \cdots & g_1^{q^{k+t-1}} & -y_1 & \cdots & -y_1^{q^t} \\ g_2 & \cdots & g_2^{q^{k+t-1}} & -y_2 & \cdots & -y_2^{q^t} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ g_{\mathrm{N}} & \cdots & g_{\mathrm{N}}^{q^{k+t-1}} & -y_{\mathrm{N}} & \cdots & -y_{\mathrm{N}}^{q^t} \end{pmatrix} \begin{pmatrix} n_1 \\ \vdots \\ n_{k+t-1} \\ v_0 \\ \vdots \\ v_t \end{pmatrix} = 0$$

L'ensemble des solutions forment un F_q^m-espace vectoriel de dimension l. Si $l = 1$, toutes les solutions du système sont des solutions de $\mathrm{R}q\mathrm{P}(y,g,k,t)$.

Algorithme de décodage des codes de Gabidulin

On suppose qu'on a reçu y, qui est égal à $c+e$, où c est un mot de C et e de rang inférieur à t.
soit (V,N) une solution du système linéarisé. Alors, on a

$$\mathrm{V}(y) = \mathrm{N}(g).$$

Donc

$$\mathrm{V}(\mathrm{P}(g)) + \mathrm{V}(e) = \mathrm{N}(g).$$

D'où

$$\mathrm{V}(e) = (\mathrm{N} - \mathrm{V} \circ \mathrm{P})(g).$$

Or, $rg(\mathrm{V}(e)) \leq rg(e) \leq \frac{n-k}{2}$, et $\mathrm{N} - \mathrm{V} \circ \mathrm{P}$, de q-degré au plus $k+t-1 \leq \frac{n+k}{2} - 1$, ne peut avoir un rang si petit. Donc, $\mathrm{N} - \mathrm{V} \circ \mathrm{P} = 0$.
N'importe quelle solution non nulle de l'équation matricielle permet alors d'obtenir P, par division euclidienne à gauche. De plus, comme une partie de S ne dépend que de g, la résolution de cette équation peut être accélérée par : $\mathrm{S} = \begin{pmatrix} \mathrm{G}_1 & \mathrm{Y}_1 \\ \mathrm{G}_2 & \mathrm{Y}_2 \end{pmatrix}$, où G_1 est une matrice carrée de taille $k+1$.
On veut alors résoudre

$$\begin{cases} \mathrm{G}_1\mathcal{N} + \mathrm{Y}_1\mathcal{V} = 0 \\ \mathrm{G}_2\mathcal{N} + \mathrm{Y}_2\mathcal{V} = 0 \end{cases}$$

Or, G_1 est inversible car les g_i sont indépendants sur $\mathbb{F}_q$. Donc, on a :

$$\begin{cases} \mathcal{N} = -\mathrm{G}_1^{-1}(\mathrm{Y}_1\mathcal{V}) \\ -\mathrm{G}_1^{-1}\mathrm{G}_2\mathrm{Y}_1\mathcal{V} + \mathrm{Y}_2\mathcal{V} = 0 \end{cases}$$

Posons $\mathrm{I} = -\mathrm{G}_1^{-1}$, $\mathrm{J} = -\mathrm{G}2\mathrm{G}_1^{-1}$, nous obtenons alors

$$\begin{cases} \mathcal{N} = \mathrm{I}(\mathrm{Y}_1\mathcal{V}) \\ (\mathrm{J}\mathrm{Y}_1 + \mathrm{Y}_2)\mathcal{V} = 0 \end{cases}$$

Après avoir fait un pré-calcul de I et J, on utilise l'algorithme Berlekamp-Welch pour le décodage :

Algorithme 3 Berlekamp-Welch

Entrée:
- y ▷ Le mot reçu
- U
- T

Sortie:
- P(X), ▷ Un q-polynôme

1: Calculer une solution $\mathcal{V}_0 \neq 0$ par multiplications matricielles et pivot de Gauss
2: Calculer $\mathrm{P} = \mathrm{V}_0 \backslash \mathrm{N}_0$, par division euclidienne à gauche
3: **si** $\mathrm{R}g(y - \mathrm{P}(g)) \leq t$ **faire**
4: $\quad$ P(X)
5: **sinon**
6: $\quad$ Échec
7: **fin si**

La complexité de cet algorithme est en $\mathrm{O}(kt^2 + t^3 + k^2)$ multiplications dans $\mathbb{F}_{q^m}$.

Théorème 2.7. *Soit* C *un code de Gabidulin de longueur n, de dimension k et de support g. Soit* $y \in \mathbb{F}_{q^m}^n$ *un mot reçu. Alors, si* $t < \frac{n-k}{2}$*, on peut décoder y dans* C *à distance t en* $\mathrm{O}(kt^2 + t^3 + k^2)$ *multiplications dans* $\mathbb{F}_{q^m}$.

2.5.3 Décodage des erreurs avec l'algorithme de Berlekamp-Massey modifié

On s'intéresse maintenant au décodage des codes de Gabidulin avec l'algorithme de *Berlekamp-Massey modifié* [RP04]. Supposons que l'erreur d'un mot de code en métrique rang est :

$$\mathrm{E} = \left(\begin{array}{cc|c|cc} 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

E c'est un erreur de rang 2. Si $t \geq 2$, on peut retrouver le mot de code.

Dans le reste de ce chapitre, on considère que C est un code de Gabidulin de langueur n, de dimension k, de distance minimale d dans le corps $\mathbb{F}_{q^m}$, de matrice de parité H. Considérons que $y = c + e$ est le mot reçu où c un mot de C, e est une erreur et r est le rang de e, avec $2r < d$.

On calcule :

$$s = y.\mathrm{H}^{\mathrm{T}} = e.\mathrm{H}^{\mathrm{T}}.$$

On définit la matrice Y à composantes dans le corps $\mathbb{F}_q$ de rang t telle que

$$e = (\mathrm{E}_0, \ldots, \mathrm{E}_{t-1})\,\mathrm{Y},$$

où $\mathrm{E}_0, \ldots, \mathrm{E}_{t-1} \in \mathbb{F}_q$ sont linéairement indépendants sous $\mathbb{F}_q$. On définit aussi la matrice Z comme suit :

$$\mathrm{Z}^{\mathrm{T}} = \mathrm{Y}\mathrm{H}^{\mathrm{T}} = \begin{pmatrix} z_0 & z_0^{[1]} & \cdots & z_0^{[d-1]} \\ \vdots & \vdots & \ddots & \vdots \\ z_{t-1} & z_{t-1}^{[1]} & \cdots & z_{t-1}^{[d-1]} \end{pmatrix}$$

Il facile de remarquer que $z_0, \ldots, z_{t-1} \in \mathbb{F}_q$ sont linéairement indépendants sous $\mathbb{F}_q$. Donc,

$$(\mathrm{S}_0, \ldots, \mathrm{S}_{d-2}) = (\mathrm{E}_0, \ldots, \mathrm{E}_{t-1})\,\mathrm{Z}^{\mathrm{T}},$$

ce qui donne,

$$\mathrm{S}_i^{[-i]} = \sum_{j=0}^{t-1} \mathrm{E}_j^{[-i]} z_j, i = 0, ..., d-2.$$

Donc, on a $d-1$ équations et $2t$ inconnues. Notons aussi que t n'est pas connu non plus. Il suffit de trouver une seule solution pour ce système pour récupérer e.
Soit $\mathrm{P}(\mathrm{X})$ un q-polynôme de q-degré t qui a comme solutions $\mathrm{E}_0, \ldots, \mathrm{E}_{t-1}$ sous $\mathbb{F}_q$ et $\mathrm{P}_0 = 1$, nous l'appelons *polynôme d'erreur*. Nous définissons aussi *l'équation clé* dans le théorème suivant :

Théorème 2.8. *Soit* P *le polynôme d'erreur, on a*

$$\mathrm{P}(\mathrm{X}) \circ \mathrm{S}(\mathrm{X}) = \mathrm{F}(\mathrm{X}) \quad mod \quad \mathrm{X}^{[d-1]},$$

où $\mathrm{S}(\mathrm{X})$ *est le polynôme linéaire du syndrome et* $\mathrm{F}(\mathrm{X})$ *est un polynôme auxiliaire avec* $deg_q(\mathrm{F}) < t$.

On doit donc résoudre le système $\sum_{j=0}^{i} \mathrm{P}_j \mathrm{S}_{i-j}^{[j]} = 0$ pour $t \leq i \leq 2t-1$ qui s'écrit encore :

$$\mathrm{S}.\begin{pmatrix} \mathrm{P}_t \\ \mathrm{P}_{t-1} \\ \vdots \\ \mathrm{P}_1 \end{pmatrix} = \begin{pmatrix} -\mathrm{S}_t \\ -\mathrm{S}_{t+1} \\ \vdots \\ -\mathrm{S}_{2t-1} \end{pmatrix}, \tag{2.9}$$

avec,

$$\mathrm{S} = \begin{pmatrix} \mathrm{S}_0^{[t]} & \cdots & \mathrm{S}_{t-1}^{[1]} \\ \mathrm{S}_1^{[t]} & \cdots & \mathrm{S}_t^{[1]} \\ \vdots & \ddots & \vdots \\ \mathrm{S}_{t-1}^{[t]} & \cdots & \mathrm{S}_{2t-1}^{[1]} \end{pmatrix}$$

On remarque que S est inversible, ce qui donne l'unicité de la solution. Cette solution peut être trouvée en utilisant l'algorithme de Berlekamp-Massey modifié.
On peut voir l'équation (2.9) comme un registre à décalage (FSR) [RP04].

Le problème de la résolution de *l'équation clé* est équivalent au problème de recherche de plus petit FSR qui génère la sequence du syndrome.
On commence les itérations par $i = 0$, on initialise la longueur du registre l_i par 0 et $\mathrm{P}(\mathrm{X}) = \mathrm{X}$. A chaque itération i, on crée un registre qui génère les $i+1$ syndromes qui ont une longueur l_i.

Définition 2.13. Le discriminant de la $i-$ème itération est défini par :

$$\Delta_i = \mathrm{S}_i + \sum_{j=1}^{l_i} \mathrm{P}_j^{(i)} \mathrm{S}_{i-j}^{[j]} = \sum_{j=0}^{l_i} \mathrm{P}_j^{(i)} \mathrm{S}_{i-j}^{[j]}$$

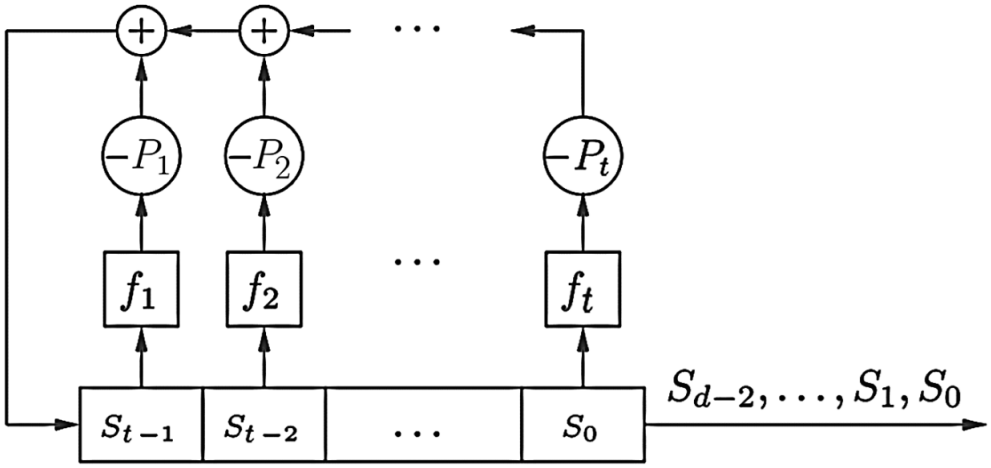


FIGURE 3 – Le polynôme d'erreur sous forme d'un registre à décalage [RP04].

Pour le cas $\Delta_i = 0$, on pose $\mathrm{P}^{(i+1)}(\mathrm{X}) = \mathrm{P}^{(i)}(\mathrm{X})$ et l'itération est terminée.

Théorème 2.9. *Si $\Delta_i \neq 0$, le polynôme linéaire est obtenu :*

$$\mathrm{P}^{(i+1)}(\mathrm{X}) = \mathrm{P}^{(i)}(\mathrm{X}) - \Delta_i \Delta_m^{-[r-m]} \circ \mathrm{P}^{(m)}(\mathrm{X}),$$

avec $m < i$. Donc, le nouveau discriminant $\Delta'_i = 0$.

Démonstration. Il suffit de vérifier que $\Delta'_i = \sum_{j=0}^{l_{i+1}} \mathrm{P}_j^{(r+1)} \mathrm{S}_{i-j}^{[j]} = 0$. □

On a prouvé que $l_{i+1} = max(l_i, r+1-l_i)$ et que l'algorithme de Berlekamp-Massey modifié pour la métrique rang génère le plus petit FSR. On peut résumer l'algorithme dans le schéma de la Figure 4.
Maintenant, on peut résumer les différentes étapes de l'algorithme de décodage :

Algorithme 4 Berlekamp-Massey modifié

Entrée:
y ▷ Le mot reçu
H ▷ La matrice de parité
Sortie:
c ▷ Le mot de code

1: Calculer le syndrome.
2: Résoudre le système **(1)** avec l'algorithme de Berlekamp-Massey modifié pour obtenir P(X).
3: Calculer $\mathrm{E}_0, \ldots, \mathrm{E}_{t-1}$ les racines de P(X).
4: Résoudre le système $\mathrm{S}_i^{[-i]} = \sum_{j=0}^{t-1} \mathrm{E}_j^{[-i]} z_j, i = 0, ..., d-2$.
5: Calculer Y par multiplications matricielles.
6: Calculer $e = (\mathrm{E}_0, \ldots, \mathrm{E}_{t-1})\mathrm{Y}$.
7: $c = y - e$.

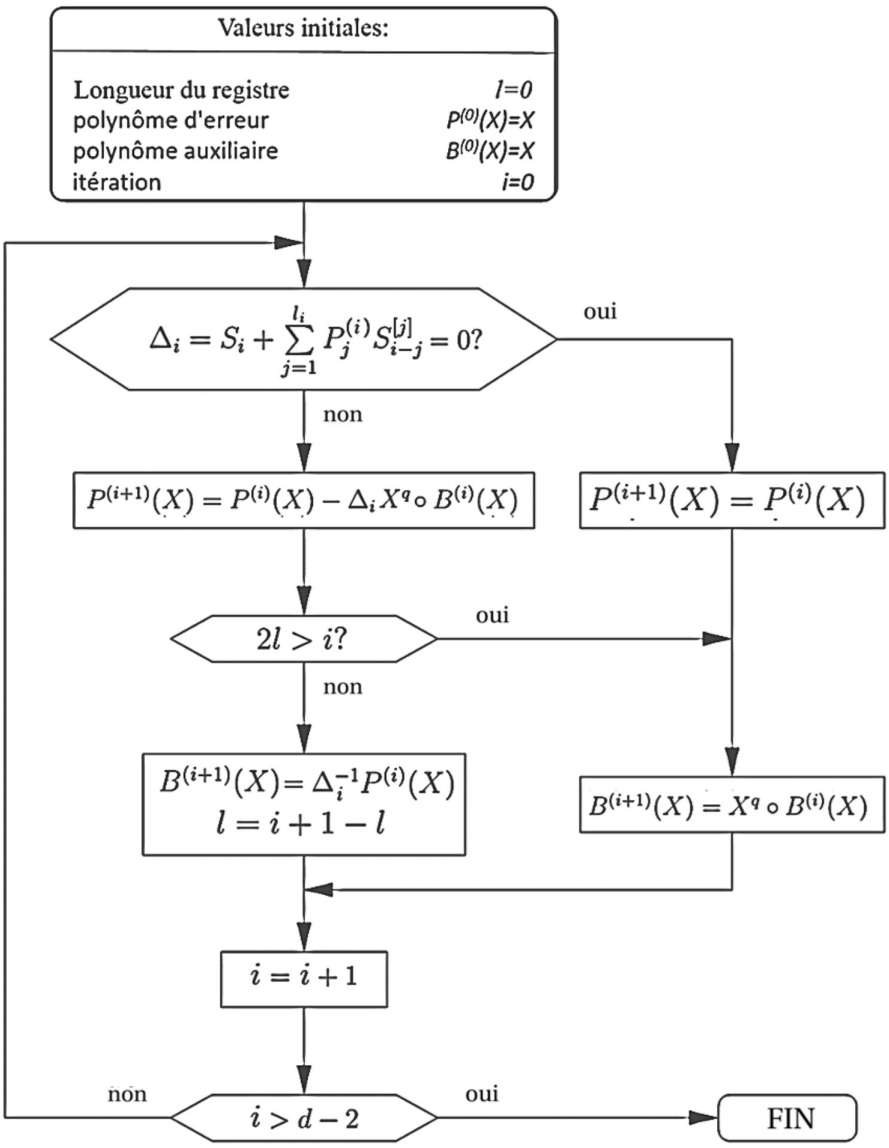


FIGURE 4 – L'algorithme de Berlekamp-Massey modifié [RP04].

2.6 Conclusion

Ce chapitre préliminaire est un rappel des propriétés principales des corps finis ainsi que des notions essentielles des codes correcteur d'erreurs. Il a été introduit dans le but d'aider le lecteur à comprendre la suite de ce rapport.

Nous avons commencé ce chapitre par présenter les propriétés principales des corps finis. Ensuite, nous avons introduit la notion des q-polynômes et nous avons présenté l'algorithme de calcul du plus grand diviseur commun à droite de deux q-polynômes que nous utiliserons dans l'algorithme de décodage des codes présentés dans le chapitre 3.

Dans la section 2.3, nous avons présenté le principe du codage et les différents type de codes qui existe. Puis, nous avons donné quelques techniques de décodage qui existe dans la littérature.

Nous avons introduit, dans la section 2.4, la métrique rang ainsi que ses principales propriétés. Nous avons présenté, par la suite, les bornes sur les codes en métrique rang (la borne de Singleton et la borne de Gilbert-Varshamov en particulier).

Dans la dernière section de ce chapitre, nous avons présenté les codes Gabidulin ainsi que leur algorithme de décodage. Ces codes appartiennent à la famille des codes optimaux, i.e. les codes Gabidulin atteignent la borne de Singleton.

Chapitre 3

Les codes LRPC

Dans le chapitre précédent, nous avons présenté des propriétés assez générales de la métrique rang. Nous avons établi les différents types de décodage. Par la suite, nous avons montré l'existence des codes optimaux en métrique rang et nous avons décrit leurs algorithmes de décodage. Dans ce chapitre, nous définissons une nouvelle famille de codes en métrique rang, qui s'appelle les codes *Low Rank Parity Check (LRPC)*.

Les codes LRPC ont été proposés par le laboratoire XLIM en 2013 [Gab+13]. Ces codes appartiennent aux familles de codes probabilistes, i.e., la capacité de correction est atteinte avec une certaine probabilité qui dépend des paramètres de code. Les codes LRPC sont considérés comme les analogues des codes LDPC dans la métrique rang. Dans leur article fondateur, P. Gaborit et al. ont présenté une nouvelle famille de codes en métrique rang, disposant d'une structure particulière et simple. Ils ont également décrit un algorithme de décodage en temps polynomial. Sa complexité est plus faible que celle des codes Gabidulin. Le principe de l'algorithme de décodage repose sur la recherche du support de l'erreur, puis, sur la récupération des coordonnées de chaque erreur. Cette approche ressemble à la méthode de décodage des codes Reed-Solomon où on calcule le polynôme localisateur de l'erreur qui donne le support, puis, le polynôme évaluateur qui donne la valeur des coordonnées de l'erreur.

Dans la première partie de ce chapitre, nous définissons les codes LRPC et nous décrivons leurs propriétés principales. Nous décrivons également l'algorithme de décodage de cette famille de codes. En suite, nous proposons des améliorations sur l'algorithme de décodage des codes LRPC et nous étudions sa complexité de décodage. En effet, nous parvenons à optimiser la complexité de décodage des codes LRPC, par rapport aux codes Gabidulin.

Dans un second temps, nous définissons les codes matriciels et nous introduisons les codes LRPC matriciels. Nous présentons les performances de ces codes par rapport aux codes Gabidulin. Ensuite, nous montrons qu'un code peut asymptotiquement atteindre la borne de Gilbert-Vershamov en métrique rang pour certains paramètres de code.

Dans la troisième partie, nous introduisons les codes LRPC étendus, qui sont une

amélioration des codes LRPC. Nous décrivons leur algorithme de décodage et nous présentons les performances de ces codes. Par la suite, nous validons les résultats théoriques par des simulations pour différents paramètres du code.

3.1 Les codes LRPC

Dans cette partie, nous définissons les codes LRPC et leurs matrices génératrices, ainsi que leurs matrices de parité comme ils sont définis dans leur article d'origine [Gab+13].

Définition 3.1. Soit $\mathcal{C}$ un code $[\mathrm{N}, k]_{q^m}$ de matrice de parité H. Le code $\mathcal{C}$ est un code LRPC de poids d si :

— La matrice H est de rang plein.

— L'espace engendré par les coefficients de la matrice H est de dimension d.

Notons F l'espace engendré par les coefficients de la matrice H et $\{\mathrm{F}_1, \mathrm{F}_2, \ldots, \mathrm{F}_d\}$ une base de F. Cela veut dire que pour tout $i \in [\![1, \mathrm{N}-k]\!]$ et $j \in [\![1, \mathrm{N}]\!]$, h_{ij} vérifie

$$h_{ij} = \sum_{v=1}^{d} h_{ijv} \mathrm{F}_v,$$

où h_{ijv} sont des éléments de $\mathbb{F}_q$.

Remarque 1. Pour les codes probabilistes, la notion de la distance minimale d'un code n'a pas de sens puisque nous pouvons décoder même au delà de cette distance avec une probabilité d'échec. Le paramètre d est le poids de la matrice H d'un code LRPC, donc il ne faut pas le confondre avec la distance minimale.

Définition 3.2. Soit $e = (e_1, e_2, \ldots, e_\mathrm{N})$ un vecteur de $\mathbb{F}_{q^m}^\mathrm{N}$ de rang r. Le sous-espace vectoriel E de $\mathbb{F}_q$ engendré par $e_1, e_2, \ldots, e_\mathrm{N}$ est appelé le support de e.

Définition 3.3. Soient E et F deux sous espaces vectoriels de $\mathbb{F}_{q^m}$. On appelle l'espace produit de E et F l'espace engendré par le produit des élément de E et de F, on le note $\langle \mathrm{EF} \rangle$.

3.1.1 Le décodage des codes LRPC

L'idée générale de l'algorithme de décodage des codes LRPC est d'utiliser le fait que le poids de la matrice H soit petit. Cela implique qu'avec une forte probabilité, l'espace S généré par le syndrome contient l'espace produit $\langle \mathrm{EF} \rangle$. Le fait d'avoir $\langle \mathrm{EF} \rangle$ et F conduit à récupérer l'espace E. Comme dans le cas des codes Gabidulin, on peut récupérer la base de l'erreur et par la suite la valeur exacte de chaque cordonnée de l'erreur e. Pour cela, il suffit de résoudre le système linéaire obtenu à partir de H et s.

Considérons $\mathcal{C}$ un code LRPC de paramètres $[\mathrm{N}, k, d]_{q^m}$, de matrice génératrice G et de matrice de parité H. Supposons que le mot reçu est $y = message \times \mathrm{G} + e$, où e est une erreur de rang r avec E le support de e. L'équation du syndrome dans $\mathbb{F}_{q^m}$ vérifie :

$$\begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k,1} & h_{n-k,2} & \cdots & h_{n-k,n} \end{pmatrix} \times \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n-k} \end{pmatrix} \tag{3.1}$$

D'abord, on calcule le support S du syndrome s. On suppose que la dimension de l'espace S est rd. Donc, S est égal à $\langle \mathrm{EF} \rangle$. En effet, pour récupérer l'espace E, il suffit de calculer l'intersection $\bigcap_{i=1}^{d} \mathrm{S}_i$ où $\mathrm{S}_i = \mathrm{F}_i^{-1}\mathrm{S}$.

Si l'intersection ne donne pas E, alors on ne peut pas récupérer le message initial. On suppose donc que $\mathrm{E} = \bigcap_{i=1}^{d} \mathrm{S}_i$. Soit $\{\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_r\}$ une base de E, on a

$$e = \sum_{j=1}^{r} e_j \mathrm{E}_j. \tag{3.2}$$

Il est possible de transformer le système d'équations $\mathbf{H}.e^{\mathrm{T}} = s$ dans $\mathbb{F}_{q^m}$ à un système d'équations dans $\mathbb{F}_q$, en exprimant les cordonnées du syndrome dans la base produit. Pour $1 \leq i \leq \mathrm{N} - k$, on a

$$s_i = \sum_{k=1}^{d} \sum_{j=1}^{r} s_{ikj} \mathrm{F}_k \mathrm{E}_j. \tag{3.3}$$

En remplaçant l'équation (3.3) dans l'équation (3.1), pour $j \in [\![1, r]\!]$, on obtient

$$\begin{cases} h_{111}e_{1j} & + & h_{121}e_{2j} & + \cdots + & h_{1\mathrm{N}1}e_{\mathrm{N}j} & = & s_{11j} \\ h_{112}e_{1j} & + & h_{122}e_{2j} & + \cdots + & h_{1\mathrm{N}2}e_{\mathrm{N}j} & = & s_{12j} \\ \vdots & & \vdots & & \vdots & & \vdots \\ h_{(\mathrm{N}-k)1d}e_{1j} & + & h_{(\mathrm{N}-k)2d}e_{2j} & + \cdots + & h_{(\mathrm{N}-k)\mathrm{N}d}e_{\mathrm{N}j} & = & s_{(\mathrm{N}-k)dj} \end{cases} \tag{3.4}$$

Pour récupérer les e_j pour $j \in [\![1, r]\!]$, il suffit de résoudre le système (3.4) dans $\mathbb{F}_q$. Ce système vérifie $\mathrm{D_H} \times e_j = s_j$, $\forall j \in [\![1, r]\!]$, où $\mathrm{D_H}$ vérifie

$$\mathbf{D}_{\mathrm{H}} = \begin{pmatrix} h_{111} & h_{121} & \cdots & h_{1\mathrm{N}1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{11d} & h_{11d} & \cdots & h_{1\mathrm{N}d} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ h_{(\mathrm{N}-k)1d} & h_{(\mathrm{N}-k)2d} & \cdots & h_{(\mathrm{N}-k)\mathrm{N}d} \end{pmatrix} \tag{3.5}$$

La matrice $\mathrm{D_H}$ ne dépend pas de l'erreur. Cela veut dire que, par un pré-calcul de l'inverse de la matrice $\mathrm{D_H}$, on peut récupérer les $e_{j=1:r}$ par un simple produit matriciel.

Pour récupérer l'erreur e, on utilise l'équation (3.2).

Pour résumer les étapes du processus de décodage des codes LRPC, que nous avons détaillées dans le paragraphe précédent, nous présentons ci-après l'algorithme de décodage des codes LRPC.

Algorithme 5 L'algorithme de décodage des codes LRPC

Entrée:
$\mathbf{H}$, ▷ la matrice de parité
y, ▷ le mot reçu
d, ▷ le poids de $\mathbf{H}$
Sortie:
x, ▷ le message

1: $s \leftarrow \mathbf{H} \times y^{\mathrm{T}}$
2: $\mathrm{S} \leftarrow < s_1, ..., s_{n-k} >$
3: **pour** $i = 1 : d$ **faire**
4: $\quad \mathrm{S}_i \leftarrow \mathrm{F}_i^{-1}\mathrm{S}$
5: **fin pour**
6: $\mathrm{E} \leftarrow \bigcap_{i=1}^{d} \mathrm{S}_i$
7: $\{\mathrm{E}_1, \mathrm{E}_2, ..., \mathrm{E}_r\} \leftarrow$ **base**(E)
8: **pour** $j = 1 : r$ **faire**
9: $\quad e_j \leftarrow$ **résoudre**$(\mathrm{D}_{\mathbf{H}}, s_j)$ ▷ $\mathrm{D}_{\mathbf{H}} \times e_j = s_j$
10: **fin pour**
11: $e \leftarrow \sum_{j=1}^{r} e_j \mathrm{E}_j$
12: $x \leftarrow$ **résoudre**$(\mathrm{G}, y - e)$ ▷ $x \times \mathrm{G} = y - e$

Le choix approprié des paramètres du code rend la probabilité d'échec de l'algorithme de décodage négligeable. Dans la pratique, il suffit de calculer $\mathrm{S}_1 \cap \mathrm{S}_2$ dans l'étape 6 pour récupérer l'espace E avec une grande probabilité.

Afin de pouvoir récupérer le maximum d'erreurs, on vérifie si S est inclus dans l'espace $\langle \mathrm{F}\tilde{\mathrm{E}} \rangle$, avec $\tilde{\mathrm{E}} = \left(\mathrm{S}_i \cap \mathrm{S}_j\right)$ pour $i \neq j$. Dans ce cas, l'espace E est récupéré entièrement. Sinon, on remplace S par $\langle \mathrm{S} + \langle \mathrm{F}\tilde{\mathrm{E}} \rangle \rangle$ et on recommence avec d'autres i et j.

Algorithme 6 L'algorithme de recherche du support E

Entrée:
S, ▷ l'espace syndrome
F
Sortie:
E, ▷ le support de l'erreur

1: $\mathrm{E} \leftarrow \mathrm{S}_1 \cap \mathrm{S}_2$
2: **tant que** $\mathrm{S} \nsubseteq \langle \mathrm{FE} \rangle$ **faire**
3: $\quad \mathrm{S} \leftarrow \mathrm{S} + \langle \mathrm{FE} \rangle$

4: $\mathrm{E} \leftarrow \mathrm{S}_i \cap \mathrm{S}_j$
5: **fin tant que**

Les étapes 3, 4, 5 et 6 de l'algorithme 5 peuvent être remplacées par l'algorithme 6.

3.1.2 Exactitude de l'algorithme de décodage

Nous étudions dans cette partie du chapitre l'exactitude des étapes de décodage de l'algorithme 5. Dans cet objectif, nous considérons trois hypothèses :

1 La dimension de l'espace $\langle \mathrm{FE} \rangle$ égale à rd.

2 L'espace $\langle \mathrm{FE} \rangle$ est inclus dans l'espace syndrome S.

3 L'intersection des S_i est incluse dans E.

Nous prouvons que, sous ces hypothèses, l'algorithme 5 décode avec succès. L'espace $\langle \mathrm{FE} \rangle$ est de dimension rd. Donc, $\{\mathrm{F}_1\mathrm{E}_1, \mathrm{F}_1\mathrm{E}_2, \ldots, \mathrm{F}_d\mathrm{E}_r\}$ forme une famille libre. Par construction, l'espace S est inclus dans $\langle \mathrm{FE} \rangle$. D'après la deuxième hypothèse, on a $\mathrm{S} = \langle \mathrm{FE} \rangle$. Donc, $\mathrm{F}_i\mathrm{E}_j$ sont tous dans S. Cela implique que E_j sont des éléments de S_i, d'où $\mathrm{E} \subset \bigcap_{i=1}^{d} \mathrm{S}_i$. D'après la troisième hypothèse, on déduit que $\mathrm{E} = \bigcap_{i=1}^{d} \mathrm{S}_i$. D'autre part, le système (3.4) a N inconnues et $(\mathrm{N} - k)\, d$ équations dans $\mathbb{F}_q$. Donc, le système admet une solution si le poids d est supérieur ou égal à $\frac{\mathrm{N}}{\mathrm{N}-k}$ et la matrice $\mathbf{D}_\mathrm{H}$ est de rang plein. Puisqu'on peut pré-calculer la matrice $\mathbf{D}_\mathrm{H}$, la dernière condition est toujours vérifiée. Finalement, l'algorithme de décodage des codes LRPC récupère le mot de code avec succès si les trois conditions citées ci-dessous sont vérifiées.

3.1.3 La probabilité de succès de décodage

Dans le paragraphe précédent, nous avons montré que l'algorithme de décodage des codes LRPC récupère le mot de code avec succès si les trois conditions sont vérifiées. On considère la probabilité de succès de décodage des codes LRPC. Celle-ci est égale au produit de trois probabilités ; la probabilité que la dimension de l'espace $\langle \mathrm{FE} \rangle$ égale rd, la probabilité que l'espace $\langle \mathrm{FE} \rangle$ soit égal à l'espace syndrome S et la probabilité que l'intersection des S_i soit égal à E. Lorsque m est plus grand que rd^2, la première et la dernière probabilités sont négligeables par rapport à la deuxième probabilité.

Proposition 3.1. *La probabilité que l'espace syndrome* S *génère l'espace produit* $\langle \mathrm{FE} \rangle$ *est plus grande que* $1 - q^{rd-(\mathrm{N}-k)}$.

3.1.4 La complexité de décodage

Dans cette partie, nous considèrerons la complexité de décodage des codes LRPC. Les étapes les plus couteuses, en terme de complexité, dans l'algorithme 5 sont l'étape 6 et l'étape 9. Pour l'étape 6, la complexité de l'intersection des espaces vectoriels est $(2rd)^2\, m$ multiplications dans le corps de base $\mathbb{F}_q$. Dans l'étape 9, la complexité de calcul peut se ramener à N^2 multiplications dans le corps $\mathbb{F}_q$, cela est possible en effectuant un pré-calcul de l'inverse de la matrice $\mathbf{D}_\mathrm{H}$. Ensuite, elle est stockée dans la

mémoire du codeur. Ce dernier calcul est effectué r fois, ce qui implique que la complexité de l'étape 9 est $\mathcal{O}\left(r\mathrm{N}^2\right)$. Le théorème suivant présente une approximation de la probabilité de décodage ainsi que la complexité de décodage des codes LRPC.

Théorème 3.1. *Soit $\mathcal{C}$ un code LRPC de paramètres $[\mathrm{N}, k, d]_{q^m}$. L'algorithme de décodage décode une erreur de rang r avec une probabilité de succès proche de $1 - q^{rd-(\mathrm{N}-k)}$ et une complexité $\mathcal{O}\left((2rd)^2 m + r\mathrm{N}^2\right)$ multiplications dans le corps de base $\mathbb{F}_q$.*

3.2 Les Codes LRPC matriciels

Les codes LRPC peuvent être intégrés dans de nombreuses applications pour protéger les données corrompues. Cependant, le codeur-décodeur LRPC consomme beaucoup de puissance et demande l'implémentation de bibliothèques supplémentaires pour le calcul dans les corps finis. L'intégration du circuit codeur et décodeur LRPC classique implique une augmentation du coût de la production de la composante matérielle à intégrer dans l'appareil. Dans cette partie nous décrivons une construction des codes LRPC basée sur des simples opérations d'algèbre linéaire.

3.2.1 Définition et propriétés

Soient p un nombre premier, q une puissance de p, m un entier positif et $\beta = \{\beta_1, \beta_2, \cdots, \beta_m\}$ une base de $\mathbb{F}_{q^m}$. Le fait de choisir $q = p$ simplifie beaucoup l'implémentation des codes LRPC dans un circuit. Nous n'avons donc pas besoin de définir des bibliothèques de calcul dans les corps finis. Pour $q = 2$, les implémentations sont encore plus simplifiées du fait que la somme *modulo q* est remplacé par des XORs.

Définition 3.4. La vectorisation est une opération linéaire qui transforme une matrice A de taille $m \times n$ en vecteur colonne v de longueur mn, elle sera notée $\mathrm{V}ect$.

La vectorisation d'une matrice A de taille $m \times n$ est

$$\mathrm{vect}\,(\mathrm{A}) = \left(a_{1,1}, a_{2,1}, \ldots, a_{m,1}, a_{m,2}, \ldots, a_{m,n}\right)^{\mathsf{T}}, \tag{3.6}$$

autrement dit $v\left(\left(j-1\right)m + i\right) = a_{i,j}$, pour $i \in [\![1, m]\!]$ et $j \in [\![1, n]\!]$.

Définition 3.5. Soient A une matrice de $\mathbb{F}_q^{m_\mathrm{A} \times n_\mathrm{A}}$ et B une matrice de $\mathbb{F}_q^{m_\mathrm{B} \times n_\mathrm{B}}$. Le produit de Kronecker de A et B est défini par

$$\mathrm{A} \otimes \mathrm{B} = \begin{pmatrix} a_{11}\mathrm{B} & a_{12}\mathrm{B} & \cdots & a_{1n_\mathrm{A}}\mathrm{B} \\ a_{21}\mathrm{B} & a_{22}\mathrm{B} & \cdots & a_{2n_\mathrm{A}}\mathrm{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_\mathrm{A}1}\mathrm{B} & a_{m_\mathrm{A}2}\mathrm{B} & \cdots & a_{m_\mathrm{A}n_\mathrm{A}}\mathrm{B} \end{pmatrix}.$$

Proposition 3.2 (La vectorisation et le produit de Kronecker).
Soient A, B *et* C *trois matrices sur $\mathbb{F}_q$, on a*

$$vect(\mathrm{ABC}^{\mathrm{T}}) = (\mathrm{C} \otimes \mathrm{A})\, vect(\mathrm{B}).$$

Nous donnons ci-après une nouvelle construction des codes LRPC conduisant aux codes LRPC matriciels dans le corps de base $\mathbb{F}_p$.

Définition 3.6. Soient $\mathrm{H}_1, \mathrm{H}_2, \ldots, \mathrm{H}_d$ d-matrices de $\mathbb{F}_q^{\mathrm{N}-k\times\mathrm{N}}$ et $\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_d$ d-matrices de $\mathbb{F}_q^{m\times m}$. Un code LRPC matriciel de paramètres $[\mathrm{N}, k, d]$ est caractérisé par sa matrice de parité H qui est définie comme la somme du produit de Kronecker de H_i et A_i

$$\mathrm{H} = \sum_{i=1}^{d} \mathrm{H}_i \otimes \mathrm{A}_i.$$

Les matrices $\mathrm{H}_1, \mathrm{H}_2, \ldots, \mathrm{H}_d$ sont choisies de telle sorte que leur concaténation ligne forme une matrice de rang N. Cela implique que $d(\mathrm{N}-k) \geq \mathrm{N}$. Les matrices $\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_d$ doivent vérifier les deux conditions suivantes :

— Les matrice $\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_d$ sont toutes inversibles.

— $\mathrm{A}_i^{-1}\mathrm{A}_j \neq \mathrm{A}_u^{-1}\mathrm{A}_v$, $\forall\,(i,v) \in [\![1,m]\!]^2$ et $\forall\,(j,u) \in [\![1,m]\!] \setminus \{i,v\}^2$.

3.2.2 Le décodage des codes LRPC matriciels

L'algorithme de décodage des codes LRPC matriciels, similairement aux codes LRPC classiques, utilise le fait que la matrice de parité H est de rang faible. On commence par la recherche du support de l'erreur par un calcul des intersection des sous-espaces vectoriels. Ensuite, à partir des équations de parité, on récupère le mot de code.

Le calcul de s peut se faire en multipliant la matrice H par le mot reçu y ou en utilisant la formule suivante

$$\mathrm{S} = \sum_{i=1}^{d} \mathrm{A}_i \mathrm{Y} \mathrm{H}_i^{\mathrm{T}}, \tag{3.7}$$

où S et Y sont respectivement les deux représentations matricielles du syndrome s et du mot reçu y vérifiant $s = \mathrm{vect}\,(\mathrm{S})$ et $y = \mathrm{vect}\,(\mathrm{Y})$. Pour ne pas confondre le support E avec la matrice erreur on note cette dernière Z.

L'étape la plus importante dans le processus de décodage est la récupération du support E. En effet, si l'algorithme de décodage des codes LRPC matriciels dispose de la base du support, le mot de code est récupéré avec succès. Cela est vrai si les paramètres du code sont bien choisis.

Nous commençons par calculer le support de la matrice syndrome S, on le note $\langle \mathrm{S} \rangle$. Considérons les trois hypothèses suivantes :

— La dimension de l'espace $\langle \mathrm{A}_1\mathrm{E}, \mathrm{A}_2\mathrm{E}, \ldots, \mathrm{A}_d\mathrm{E} \rangle$ égale rd.

— L'espace S est égal à $\langle \mathrm{A}_1\mathrm{E}, \mathrm{A}_2\mathrm{E}, \ldots, \mathrm{A}_d\mathrm{E} \rangle$.

— L'intersection des $\mathrm{A}_i^{-1}\mathrm{S}$ est égale à E.

Les colonnes de la matrice S sont incluses dans l'espace généré par $\langle A_1E, A_2E, \ldots, A_dE\rangle$. Par conséquent, E Peut être récupéré en calculant $\bigcap_{i=1}^{d} A_i^{-1}S$. L'intersection des matrices A_iS se fait en utilisant le pivot de Gauss. Le principe est simple, on commence par initialiser la matrice de calcul S_{pv} :

$$S_{pv} = \left[\begin{array}{cccc} A_1^{-1}S & A_2^{-1}S & \cdots & A_d^{-1}S \\ \hline I_{N-k} & & & \\ & I_{N-k} & & \\ & & \ddots & \\ & & & I_{N-k} \end{array}\right]$$

Après l'application du pivot de Gauss, on obtient :

$$S_{BU} = \begin{bmatrix} S_B & 0 \\ B_1 & U_1 \\ \vdots & \vdots \\ B_d & U_d \end{bmatrix}$$

où la matrice S_B forme une base de l'espace $\langle A_1^{-1}S, A_2^{-1}S, \ldots, A_d^{-1}S\rangle$ et les matrices $U_{i=1:d}$ s'appellent les matrices des indices. On peut donc récupérer une base de E en prenant les combinaisons de S_{BU} qui forment des vecteurs colonnes nuls dans les premiers m coefficients, c'est à dire que $E = \langle A_1^{-1}SU_1^{\intercal}, \ldots, A_d^{-1}SU_d^{\intercal}\rangle$.

Le coût de l'étape de recherche du support E est r^2d^4m multiplications dans le corps $\mathbb{F}_q$. En général, pour retrouver la base E avec une grande probabilité, il suffit de calculer l'intersection entre $A_1^{-1}S$ et $A_2^{-1}S$. Le coût de cette étape dans ce cas est $(2rd)^2\, m$ multiplications dans le corps $\mathbb{F}_p$.

Finalement, on exprime les coordonnées de e dans une base de E en résolvant un système linéaire. Cela peut être fait en utilisant deux méthodes. La première méthode est l'utilisation l'équation (3.4) de l'algorithme de décodage des codes LRPC classiques. La deuxième méthode est l'utilisation la proposition 3.2 et la décomposition de l'erreur en deux matrices : B la matrice support et D la matrice des positions respectivement de taille $m \times r$ et $r \times N$. Nous aurons besoin de cette deuxième méthode dans le chapitre 4. D'après la proposition 3.2, on a :

$$s = \text{vect}\left(\sum_{i=1}^{d} A_iZH_i^{\intercal}\right) = \left(\sum_{i=1}^{d} H_i \otimes A_i\right) z.$$

Par conséquent,

$$s = \text{vect}\left(\sum_{i=1}^{d} A_iBDH_i^{\intercal}\right) = \left(\sum_{i=1}^{d} H_i \otimes A_iB\right) z_D. \tag{3.8}$$

La matrice $\sum_{i=1}^{d} H_i \otimes A_iB$ est de taille $(N-k)\, m \times rN$. Donc, le système (3.8) ne peut admettre une solution unique que si $r \leq \left\lfloor \left(1 - \frac{k}{N}\right) m \right\rfloor$. Cette condition est toujours

vérifiée puisque le support de l'erreur ne peut pas être récupéré si r est au-dessous de cette valeur.

Proposition 3.3. *La matrice* $\sum_{i=1}^{d} \mathrm{H}_i \otimes \mathrm{A}_i\mathrm{B}$ *est de rang* $r\mathrm{N}$.

Démonstration. Considérons la matrice définie ci-dessous $\mathrm{H}^{(\mathrm{B})} = \sum_{i=1}^{d} \mathrm{H}_i \otimes \mathrm{A}_i\mathrm{B}$. Les paramètres d et r vérifient $d(\mathrm{N}-k) \geq \mathrm{N}$ et $m \geq rd$. Soit v un vecteur de $\mathbb{F}_p^{r\mathrm{N}}$ vérifiant $\mathrm{H}^{(\mathrm{B})} v^{\intercal} = 0$. Alors, on a :

$$\begin{pmatrix} \sum_{i=1}^{d} h_{i11}\mathrm{B}_i & \sum_{i=1}^{d} h_{i12}\mathrm{B}_i & \cdots & \sum_{i=1}^{d} h_{i1\mathrm{N}}\mathrm{B}_i \\ \sum_{i=1}^{d} h_{i21}\mathrm{B}_i & \sum_{i=1}^{d} h_{i22}\mathrm{B}_i & \cdots & \sum_{i=1}^{d} h_{i2\mathrm{N}}\mathrm{B}_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d} h_{i(\mathrm{N}-k)1}\mathrm{B}_i & \sum_{i=1}^{d} h_{i(\mathrm{N}-k)2}\mathrm{B}_i & \cdots & \sum_{i=1}^{d} h_{i(\mathrm{N}-k)\mathrm{N}}\mathrm{B}_i \end{pmatrix} \begin{pmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{r1} \\ v_{12} \\ \vdots \\ h_{r\mathrm{N}} \end{pmatrix} = 0$$

D'après les suppositions du paragraphe 3.2.2, les colonnes des matrices $\mathrm{B}_i = \mathrm{A}_i\mathrm{B}$ forment une famille libre, donc

$$\begin{cases} h_{111}v_{11} & + & h_{112}v_{12} & +\cdots+ & h_{11\mathrm{N}}v_{1\mathrm{N}} & = & 0 \\ h_{111}v_{21} & + & h_{112}v_{22} & +\cdots+ & h_{11\mathrm{N}}v_{2\mathrm{N}} & = & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ h_{111}v_{r1} & + & h_{112}v_{r2} & +\cdots+ & h_{11\mathrm{N}}v_{r\mathrm{N}} & = & 0 \\ h_{211}v_{11} & + & h_{212}v_{12} & +\cdots+ & h_{21\mathrm{N}}v_{1\mathrm{N}} & = & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ h_{d(\mathrm{N}-k)1}v_{r1} & + & h_{d(\mathrm{N}-k)2}v_{r2} & +\cdots+ & h_{d(\mathrm{N}-k)\mathrm{N}}e_{r\mathrm{N}} & = & 0 \end{cases}$$

Puisque la concaténation des matrices H_i est de rang N, les colonnes de cette matrice forment une famille libre. Par conséquent, le vecteur v est nul. □

On déduit de la proposition 3.3 que, si on récupère le support de l'erreur E, on peut toujours décoder.

Le coût de récupération des positions de l'erreur en utilisant l'équation (3.4) est le même que celui des codes LRPC classiques, i.e. $r\mathrm{N}^2$ multiplications dans le corps $\mathbb{F}_p$. Dans le deuxième cas, la complexité de calcul est la complexité d'une inversion matricielle qui est $\mathcal{O}\left((r\mathrm{N})^3\right)$. Le coût total de décodage des codes LRPC matriciel est $\mathcal{O}\left(mr^2 + r\mathrm{N}^2\right)$.

3.2.3 Comparaison des codes LRPC matriciels et des codes Gabidulin

Dans cette partie, nous comparons les performances des codes LRPC matriciel avec les performances des codes Gabidulin. En termes de capacité de décodage pure, les codes LRPC matriciels peuvent atteindre $\frac{\mathrm{N}-k}{2}$ pour $d = 2$ et la capacité de décodage

des codes Gabidulin est $\left\lfloor \frac{N-k}{2} \right\rfloor$. Ceci donne un avantage aux codes LRPC sur les codes Gabidulin en terme de complexité de calcul. Par contre, en terme de décodage, les codes Gabidulin sont plus performants.

Pour comparer les codes Gabidulin et les codes LRPC, il faut que le rendement des deux codes soit le même et qu'ils aient les mêmes longueurs de mots de codes. La capacité de correction des codes Gabidulin est maximale quand $m = N$. Nous essayons donc de trouver le meilleur choix des paramètres pour avoir une capacité de correction maximale pour les codes LRPC matriciels.

Pour ne pas confondre les notations des deux codes, nous considérons les paramètres m_g, N_g et k_g pour les codes Gabidulin et m_l, N_l et k_l pour les codes LRPC matriciels. La capacité théorique de correction d'un code LRPC est maximale lorsque $\frac{N_l - k_l}{d} = \frac{m_l}{d^2-d+1}$ (voir section 3). On pose,

$$\tau = \frac{k_l}{N_l} = \frac{k_g}{N_g}.$$

On a,

$$N_l m_l = N_g m_g \Leftrightarrow N_l = \sqrt{\frac{d}{(1-\tau)(d^2-d+1)}} N_g.$$

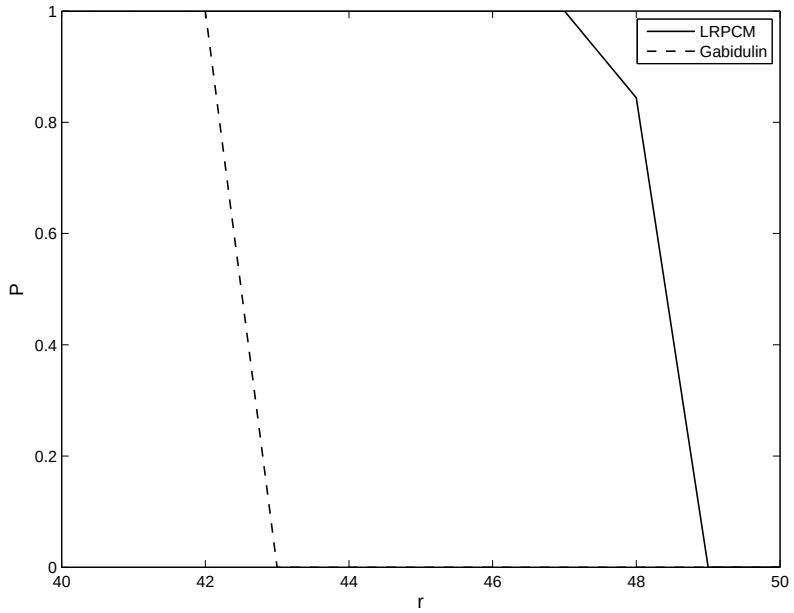


FIGURE 1 – Probabilité de décodage des codes Gabidulin et des codes LRPC matriciels pour $m_g = N_g = 168$, $N_l = 196$, $m_l = 144$ et $\tau = 0.5$.

Donc, la capacité théorique de correction d'un code LRPC matriciel est :

$$t_l = \sqrt{\frac{d}{(1-\tau)(d^2-d+1)}} t_g,$$

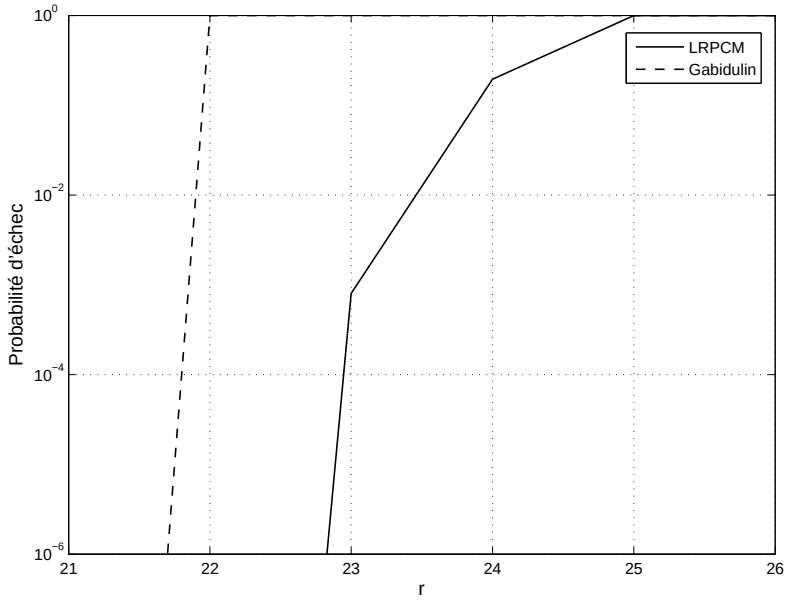


FIGURE 2 – Probabilité d'échec de décodage des codes Gabidulin et des codes LRPC matriciels pour $m_g = \mathrm{N}_g = 84$, $\mathrm{N}_l = 98$, $m_l = 72$ et $\tau = 0.5$.

où t_g est la capacité de correction de Gabidulin. Pour $\tau = \frac{1}{2}$ et $d = 2$ on a,

$$t_l \simeq 1.15\, t_g = 0.29 \times \mathrm{N}_g \tag{3.9}$$

Notons que la capacité théorique de correction des codes LRPC matriciels t_l est la valeur maximale puisque les LRPC matriciels sont des codes probabilistes.

Dans cette partie, nous étudions, à travers des simulations, le comportement des Codes LRPC matriciels et des codes Gabidulin. Dans les simulations, nous respectons le choix des paramètres définis ci-dessous.

La Figure 1 présente les performances du code LRPC matriciel, en terme de capacité de décodage, par rapport au code Gabidulin en présence de r erreurs rang. Nous utilisons le corps de base $\mathbb{F}_7$. Nous comparons ainsi un code LRPC matriciel de paramètres $m_l = 144$, $\mathrm{N}_l = 196$ et $k_l = 98$ avec un code Gabidulin de paramètres $m_g = 168$, $\mathrm{N}_g = 168$ et $k_g = 84$. Notons que les deux codes sont de tailles égales. On remarque que les codes LRPC matriciels sont plus performants que les codes Gabidulin. Ces résultats sont prévisibles dès lors que nous avons utilisé les paramètres de code LRPC matriciel qui vérifient (3.9). Par conséquents, les codes LRPC matriciels ont 15% de gain par rapport aux codes Gabidulin.

Dans la Figure 2, la comparaison est effectuée entre un code LRPC matriciel de paramètres $m_l = 72$, $\mathrm{N}_l = 98$ et $k_l = 49$ et un code Gabidulin de paramètres $m_l = 84$, $\mathrm{N}_l = 84$ et $k_l = 42$. Cette fois, nous comparons les deux codes en terme de probabilité d'échec. Nous avons choisi une échelle logarithmique pour pouvoir visualiser le

comportement des deux codes pour des valeurs très faible de probabilité d'échec. On remarque que les codes LRPC sont toujours performants par rapport au code Gabidulin. L'écart entre les deux courbes est devenu plus serré, ce qui est normal puisque l'algorithme de décodage des codes Gabidulin est déterministe. De plus, nous avons utilisé des tailles de codes plus petites. Pour les codes LRPC matriciels, la probabilité d'échec tends exponentiellement vers 0 car nous avons considéré le corps de base $\mathbb{F}_7$ plutôt que $\mathbb{F}_2$.

3.2.4 Les codes LRPC matriciels et la borne de Gilbert-Varshamov

Dans le chapitre 2, nous avons défini la borne de Gilbert-Varshamov pour la métrique rang. On prend $d = 2$, on a $\frac{\mathrm{N}_l - k_l}{2} = \frac{m_l}{3}$. D'après l'équation (2.7), on déduit

$$\begin{aligned}\mathrm{RGV}\,(m_l, k_l, \mathrm{N}_l) &\sim \frac{m_l + \mathrm{N}_l - \sqrt{(m_l - \mathrm{N}_l)^2 + 4k_l m_l}}{2} \\ &\sim \frac{\frac{3}{2}(1-\tau) - \sqrt{\frac{9}{4}(1-\tau)^2 + 6\tau(1-\tau)}}{2}\mathrm{N}_l\end{aligned}$$

Pour un rendement τ égale $\frac{1}{2}$, on a

$$\mathrm{RGV}\,(\mathrm{N}_l) \sim \frac{\mathrm{N}_l}{4} \sim t_l. \tag{3.10}$$

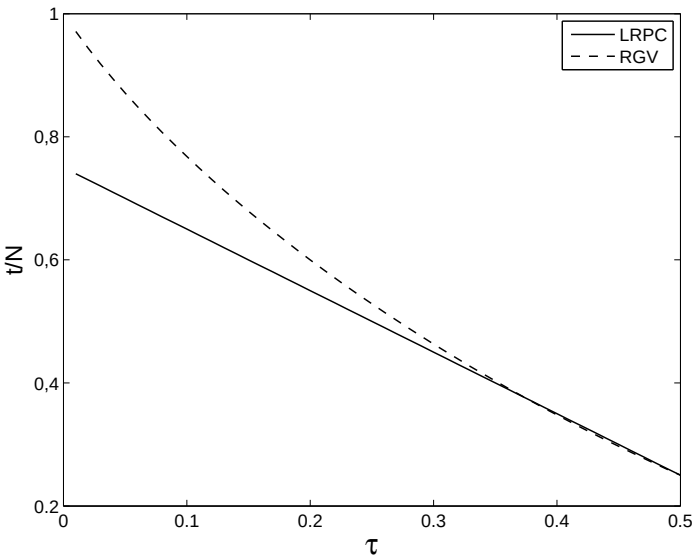


FIGURE 3 – La borne RGV comparée à la capacité théorique de correction des codes LRPC en fonction du rendement de code τ.

La Figure 3 illustre le comportement de la capacité théorique de correction des codes LRPC matriciels par rapport à la borne Gilbert-Varshamov en fonction du rendement du code. On remarque que, pour un rendement supérieure à 0.35, la capacité

théorique de correction est très proche de la borne RGV. On déduit que les codes LRPC sont plus efficaces lorsque $0.35 \lessapprox \tau \leq 0.5$.

3.3 Les codes LRPC étendus

Les codes LRPC étendus sont une amélioration des codes LRPC classique. L'idée principale est qu'au lieu d'utiliser un mot de code contenant le message initiale avec une redondance, on ajoute un vecteur nul ce qui implique qu'une partie de la matrice génératrice est nulle. Dans ce cas, on possède une information de plus sur l'erreur et plus précisément sur le support de l'erreur sans avoir besoin de calculer l'intersection. Cela nous permet d'avoir une grande capacité de correction par rapport aux codes LRPC classique.

3.3.1 Définitions et propriétés

Définition 3.7 (Code LRPC étendu).
Soit $\mathcal{C}$ un code $[\mathrm{N}, k]_{q^m}$ de matrice de parité H. $\mathcal{C}$ est un code LRPC étendu de paramètres $[\mathrm{N}, k, d, t]$ si :

— La matrice H est de rang plein.

— Les coefficients h_{ij} sont tous dans $\mathbb{F}_q$, pour $i = t+1 \ldots \mathrm{N}-k$ et $j = 1 \ldots \mathrm{N}$.

— Les coefficients h_{ij} sont tous nuls, pour $i = 1 \ldots t$ et $j = \mathrm{N}-t+1 \ldots \mathrm{N}$.

— L'espace engendré par les coefficients h_{ij} est de dimension d, pour $i = 1 \ldots t$ et $j = 1 \ldots \mathrm{N}-t$.

La matrice de parité d'un code LRPC étendu se compose de trois matrices $\mathrm{H}_{\mathrm{LRPC}}$, $\mathrm{A}_{\mathbb{F}_q}$ et $\mathrm{B}_{\mathbb{F}_q}$ respectivement de taille $(\mathrm{N}-k-t) \times (\mathrm{N}-t)$, $t \times (\mathrm{N}-t)$ et $t \times t$. Les matrices $\mathrm{A}_{\mathbb{F}_q}$ et $\mathrm{B}_{\mathbb{F}_q}$ sont à coefficients dans le corps de base et $\mathrm{H}_{\mathrm{LRPC}}$ est une matrice de parité d'un code LRPC classique de paramètres $[\mathrm{N}-t, k, d]$. La matrice H est de la forme

$$\mathrm{H} = \begin{pmatrix} \mathrm{H}_{\mathrm{LRPC}} & 0 \\ \mathrm{A}_{\mathbb{F}_q} & \mathrm{B}_{\mathbb{F}_q} \end{pmatrix}. \tag{3.11}$$

La matrice H est écrite sous la forme standard si $\mathrm{A}_{\mathbb{F}_q}$ est nulle et $\mathrm{B}_{\mathbb{F}_q}$ est égale à la matrice identité

$$\mathrm{H} = \begin{pmatrix} \mathrm{H}_{\mathrm{LRPC}} & 0 \\ 0 & \mathrm{I}_t \end{pmatrix}. \tag{3.12}$$

3.3.2 Le décodage des codes LRPC étendus

Le principe de l'algorithme de décodage des codes LRPC étendus est le même que celui des codes LRPC classiques. La seule différence dans les deux algorithmes est l'initialisation du support de l'erreur. Dans l'algorithme de décodage des codes LRPC étendus, le support de l'erreur contient déjà t-éléments de l'erreur qui ne forment pas forcement une famille libre sur $\mathbb{F}_q$. Lorsque t est égale à 0, les codes LRPC étendus sont

des codes LRPC classiques. On peut donc considérer les codes LRPC étendus comme une généralisation des codes LRPC classiques ou des codes LRPC matriciels.

Algorithme 7 L'algorithme de décodage des codes LRPC étendus

Entrée:
- $\mathbf{H}$, ▷ la matrice de parité
- y, ▷ le mot reçu
- d, ▷ le poids de $\mathbf{H}$

Sortie:
- x, ▷ le message récupéré

1: $s \leftarrow \mathbf{H} \times y^{\mathrm{T}}$
2: $\mathrm{S} \leftarrow \langle s_1, ..., s_{\mathrm{N}-k-t} \rangle$
3: $\mathrm{E} \leftarrow \langle s_{\mathrm{N}-k-t+1}, ..., s_{\mathrm{N}-k} \rangle$
4: **si** $\mathrm{S} \nsubseteq \langle \mathrm{FE} \rangle$ **faire**
5: $\quad \mathrm{S} \leftarrow \langle \mathrm{FE} \rangle + \mathrm{S}$
6: $\quad$ **pour** $i := 1 : d$ **faire**
7: $\quad\quad \mathrm{S}_i \leftarrow \mathrm{F}_i^{-1}\mathrm{S}$
8: $\quad$ **fin pour**
9: $\quad \mathrm{E} \leftarrow \mathrm{S}_1 \cap \mathrm{S}_2 \cap \ldots \cap \mathrm{S}_d$
10: **fin si**
11: $\{\mathrm{E}_1, \mathrm{E}_2, ..., \mathrm{E}_r\} \leftarrow$ **base**(E)
12: **pour** $j := 1 : r$ **faire**
13: $\quad e_j \leftarrow$ **résoudre**$(\mathrm{D}_{\mathbf{H}}, s_j)$ ▷ $\mathrm{D}_{\mathbf{H}} \times e_j = s_j$
14: **fin pour**
15: $e \leftarrow \sum_{j=1}^{r} e_j \mathrm{E}_j$

3.3.3 La probabilité de décodage des codes LRPC étendus

Nous avons besoin d'étudier la probabilité de récupérer avec succès la base de l'erreur E dans le but d'estimer la probabilité de succès de l'algorithme de décodage. Par construction, les t derniers éléments du syndrome appartiennent à E. Ces éléments ne forment pas forcement une familles libre sur $\mathbb{F}_q$. Il faut donc chercher d'autres éléments dans le syndrome pour former une base de E. Rappelons que les codes LRPC étendus sont une généralisation des codes LRPC classiques ou matriciels. Pour cette raison, nous avons choisi de calculer les probabilités de décodage dans cette partie du chapitre.

Rappelons qu'on peut récupérer le mot de code en utilisant l'algorithme de décodage des LRPC présenté ci-dessous si :

1 La dimension de l'espace $\langle \mathrm{FE} \rangle$ est égale à rd.

2 L'espace $\langle \mathrm{FE} \rangle$ est égal à l'espace S.

3 L'intersection des S_i est égale à E.

Pour chacune de ces conditions, nous proposons la probabilité pour laquelle cette condition est vérifiée.

Dans ce qui suit, sauf si c'est mentionné, nous utilisons un code LRPC pour designer un code LRPC matriciel. Dans la proposition suivante, nous avons besoin de la définition 3.5 du produit de Kronecker. Soit A une matrice de $\mathbb{F}_q^{t\times r}$, B une matrice $\mathbb{F}_q^{n\times r'}$ et C une matrice définie comme la concaténation des lignes de la matrice $\mathrm{I}_d \otimes \mathrm{A}$ et de la matrice B telles que I_d est la matrice d'identité de taille $d \times d$ et r' est égal à dr.

Proposition 3.4. *La probabilité que la matrice* C *soit de rang* rd *est donnée par*

$$\mathrm{P_S} = \sum_{i=0}^{t} \frac{\mathrm{S}\left(t,r,q,i\right)}{q^{rt}} \times \prod_{j=0}^{d(r-i)-1} \left(1-q^{j-n}\right).$$

Démonstration. Soient A et B des matrices respectivement de taille $t \times r$ et $n \times r'$ sur $\mathbb{F}_q$. On définit la matrice C comme suit

$$\mathrm{C} = \left[\begin{array}{ccc} \mathrm{A} & & \\ & \ddots & \\ & & \mathrm{A} \\ \hline & \mathrm{B} & \end{array}\right]$$

Si le rang de la matrice A est égal à i, le rang de la matrice $\mathrm{I}_d \otimes \mathrm{A}$ égale à di puisque A est répétée d-fois. Cela veut dire que les $d(r-i)$ colonnes de de la matrice B correspondant aux colonnes liés de la matrice $\mathrm{I}_d \otimes \mathrm{A}$ doivent former une base. Dans ce cas, la matrice C est de rang rd avec une probabilité égale à $\prod_{j=0}^{d(r-i)-1} \left(1-q^{j-n}\right)$.

La matrice A est de rang i avec une probabilité : $\dfrac{\mathrm{S}\left(t,r,q,i\right)}{q^{rt}}$. □

La proposition précédente donne la probabilité pour que le syndrome contienne tous les éléments $\left\{\mathrm{A}_i\mathrm{E}_j\right\}$, sans qu'ils soient forcement linéairement indépendants, sachant qu'on dispose de t éléments de support. Dans la proposition suivante, nous calculons la probabilité pour que les éléments $\left\{\mathrm{A}_i\mathrm{E}_j\right\}$ soient linéairement indépendants.

Proposition 3.5. *Soient* $\{\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_d\}$ d *matrices inversibles de tailles* $m \times m$ *sur* $\mathbb{F}_q$ *et* $\{\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_r\}$ r *vecteurs de* $\mathbb{F}_q^m$ *formant une famille libre. La probabilité que* $\left\{\mathrm{A}_i\mathrm{E}_j\right\}$ *forment une famille libre sur* $\mathbb{F}_q$*, pour* $i \in [\![1,d]\!]$ *et* $k \in [\![1,r]\!]$*, vérifie*

$$\mathrm{P_{AE}} = \prod_{j=0}^{dr-1} \left(1-q^{j-m}\right).$$

Démonstration. La probabilité que les vecteurs $\mathrm{A}_i\mathrm{E}_j$ soient linéairement indépendants est égale à la probabilité que dr vecteurs tirés aléatoirement de $\mathbb{F}_q^m$ soient linéairement indépendants, ce qui est équivalent à trouver la probabilité qu'une matrice de taille $r' \times m$ soit de rang r', où $r' = dr$. Le nombre de matrice de rang r' est calculé dans

dans l'équation (2.4) du chapitre 2. La probabilité qu'une matrice de taille $r' \times m$ soit de rang r' vérifie

$$\mathrm{P}\left(m, r', q\right) = \prod_{i=0}^{r'-1} \left(1 - q^{i-m}\right).$$

□

La probabilité calculée prend en considération même le cas où les vecteurs de la matrice sont nulles. Puisque cela arrive avec une faible probabilité, nous avons considéré ce cas pour simplifier les calculs. Dans la proposition suivante, nous calculons la probabilité pour que l'intersection des S_i ne contienne pas de vecteurs autres que des vecteurs de E.

Proposition 3.6. *Soient* $\{\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_d\}$ *d matrices inversibles de tailles $m \times m$ sur $\mathbb{F}_q$ et* $\{\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_r\}$ *r vecteurs de $\mathbb{F}_q^m$ formant une famille libre. La probabilité que* $\left\{\mathrm{A}_i^{-1}\mathrm{A}_j\mathrm{E}_k\right\}$ *forment une famille libre sur $\mathbb{F}_q$, pour $(i, j) \in [\![1, d]\!]^2$ et $k \in [\![1, r]\!]$, vérifie*

$$\mathrm{P}_{\mathrm{AAE}}\left(m, r, q, d\right) = \prod_{j=0}^{r\left(d^2-d-1\right)-1} \left(1 - q^{j-m}\right).$$

Démonstration. Remarquons que pour $(i, j) \in [\![1, d]\!]^2$, $\mathrm{A}_i^{-1}\mathrm{A}_j$ est égale à la matrice identité d-fois. Cela veut dire que dr vecteurs forment un espace de dimension r. Par conséquent, pour $(i, j) \in [\![1, d]\!]^2$ et $k \in [\![1, r]\!]$, les vecteurs $\left\{\mathrm{A}_i^{-1}\mathrm{A}_j\mathrm{E}_k\right\}$ forment un espace de dimension $rd^2 - rd + r$ au maximum. En utilisant la proposition 3.5 et en remplaçant dr par $r\left(d^2 - d - 1\right)$, on déduit le résultat. □

A partir de la proposition 3.4, de la proposition 3.5 et de la proposition 3.6, on déduit l'expression générale de la probabilité de décodage.

Théorème 3.2. *Soit $\mathcal{C}$ un code LRPC étendu de paramètre* $[\mathrm{N}, k, d, t]$ *sur $\mathbb{F}_{q^m}$. Soit e une erreur de rang r, on peut décoder e avec une probabilité :*

$$\mathrm{P}_{\mathrm{R}} = \sum_{i=0}^{t} \frac{\mathrm{S}\left(t, r, q, i\right)}{q^{rt}} \times \prod_{j=0}^{d(r-i)-1} \left(1 - q^{j-(\mathrm{N}-k-t)}\right) \times \prod_{j=0}^{dr-1} \left(1 - q^{j-m}\right) \times \prod_{j=0}^{r\left(d^2-d-1\right)-1} \left(1 - q^{j-m}\right).$$

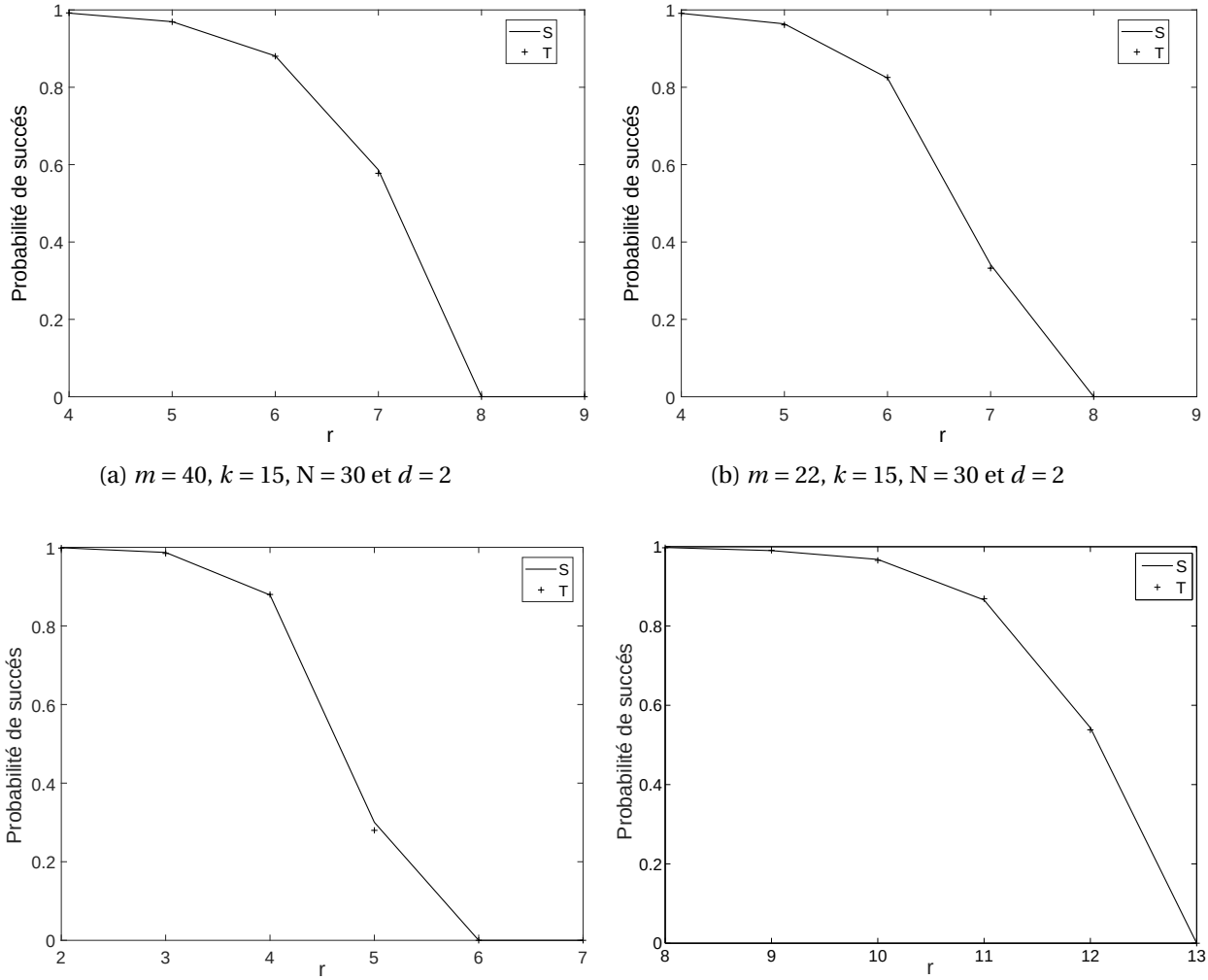
Démonstration. Le résultat est immédiat dès lors qu'on sait que le l'algorithme de décodage récupère le mot de code envoyé avec succès sous les trois conditions susmentionnées. Alors, on a

$$\mathrm{P}_{\mathrm{R}} = \mathrm{P}_{\mathrm{S}} \times \mathrm{P}_{\mathrm{AE}} \times \mathrm{P}_{\mathrm{AAE}}.$$

□

Pour m assez grand, le rang de l'erreur r peut aller jusqu'à $\left\lfloor \frac{\mathrm{N}-k-t}{d} + t \right\rfloor$. La complexité du calcul est de même grandeur que celle de l'algorithme de décodage des codes LRPC classiques.

La Figure 4 présente le résultat de simulation de la probabilité de décodage en fonction du rang de l'erreur pour différentes paramètres d'un code LRPC. La lettre S désigne la courbe de simulation de la probabilité de décodage et la lettre T désigne la courbe



(a) $m = 40$, $k = 15$, N = 30 et $d = 2$

(b) $m = 22$, $k = 15$, N = 30 et $d = 2$

(c) $m = 40$, $k = 15$, N = 30 et $d = 3$

(d) $m = 40$, $k = 15$, N = 34, $d = 2$ et $t = 4$

FIGURE 4 – La probabilité de décodage en fonction du rang de l'erreur pour différentes valeurs.

théorique de la probabilité de décodage calculée dans le théorème 3.2. On remarque que les résultats des simulations et les résultat théoriques sont très proches pour différent choix des paramètres, ce qui confirme les résultats du théorème 3.2. Les paramètres choisis dans la Figure 4a et la Figure 4c sont $m = 40$, $k = 15$, N = 30 et $d = 2, 3$ et $t = 0$. L'algorithme de décodage est capable dans ce cas de récupérer une erreur de rang $\left\lfloor \frac{N-k}{d} \right\rfloor = 7$ au maximum. Pour vérifier l'exactitude de la probabilité P_{AAE}, on a choisit des paramètres de code pour que sa valeur dans l'expression P_R ne soit pas négligeable. Pour cela, il suffit de fixer m à 22. On remarque que, dans la Figure 4b, les deux courbes sont très proches. Cela implique que l'expression P_{AAE}, qui n'est pas négligeable dans ce cas, est exacte. Dans la Figure 4d, quatre éléments du support d'erreurs sont supposés connus. L'algorithme de décodage est capable de récupérer dans ce cas une erreur de rang $\left\lfloor \frac{N-k-t}{d} \right\rfloor + t = 12$ au maximum.

3.4 Conclusion

Nous avons introduit dans ce chapitre les codes LRPC qui sont proposés à la base pour des applications dans le domaine de la cryptographie. Ces codes appartiennent à la famille des codes probabilistes. Par conséquent, nous avons défini la probabilité de décodage des codes LRPC ainsi que la complexité de leur algorithme de décodage.

Nous avons remarqué que l'algorithme de décodage des codes LRPC est moins complexe que celui des codes Gabidulin tandis que ce dernier est déterministe. Plus précisément, les codes Gabidulin sont plus performants que les codes LRPC en termes de la capacité de correction. Nous avons donc proposé les codes LRPC matriciels qui sont une transformation des codes LRPC classiques. En effet, nous avons utilisé des opérations dans un espace vectoriel et non pas dans un corps fini, cela permet donc de réduire la complexité circuit. Nous avons étudié asymptotiquement la capacité de correction de la famille de codes LRPC matriciels pour différents paramètres. Nous avons montré que pour certains paramètres les codes LRPC peuvent atteindre la borne Gilbert-Varshamov en métrique rang. Nous avons montré à travers des simulations que les codes LRPC matriciels sont plus performants que les codes Gabidulin en termes de la capacité de correction et de la complexité.

Nous avons présenté, dans la dernière section, les codes LRPC étendus qui sont une amélioration des codes LRPC. Par ailleurs, nous avons proposé une analyse théorique de la probabilité de décodage pour ces types de codes. L'expression que nous avons obtenue est une généralisation de la probabilité de décodage des codes LRPC. Nous avons montré, à travers des simulations, que la probabilité de décodage calculée analytiquement pour les codes LRPC étendus est exacte.

Chapitre 4

Codage réseau

Le codage réseau, Netwwork Coding (NC), a été introduit par Ahlswede et al. dans l'article [Ahl+00] où la technique du routage *store and forward* a été remise en question. Contrairement à la technique du routage qui considère l'information dans les paquets comme une entité immuable sans y faire aucun traitement, la technique du codage réseau utilise des méthodes algébriques pour combiner les paquets de données entre eux afin d'améliorer les performances du réseau. Ce processus est répété par chaque nœud jusqu'à ce que les paquets sources atteignent leurs destinations. Cette solution permet d'améliorer considérablement l'efficacité du réseau en réduisant le nombre de paquets qui y circulent. Ainsi, cette technique est une solution appropriée pour augmenter le débit de données et pour réduire la consommation d'énergie pour les réseaux filaires et sans fil.

L'amélioration des performances d'un réseau utilisant le codage réseau dépend du nombre de paquets envoyés par la source, de la façon dont ils sont combinés entre eux et des informations dont dispose la destination sur la topologie et sur l'état du réseau pour pouvoir effectuer le décodage adéquat.

Dans l'article d'origine [Ahl+00], le codage réseau est défini comme une approche théorique qui n'est applicable que dans les réseaux filaires. Son utilisation nécessitait une autorité centralisée pour gérer les transmissions et pour fournir les paramètres du réseau aux destinations. Ce n'est que dans leurs article [Ho+03] qui a été étendu dans [Ho+06], que Ho, Médard et Kötter ont proposé le codage réseau. Le codage réseau aléatoire, dans son nouveau nom, est devenu applicable dans la pratique en raison de sa structure aléatoire qui ne dépend pas de la topologie du réseau.

Le codage réseau permet d'améliorer les performances du réseau en termes de débit car son utilisation permet au réseau d'atteindre sa capacité maximale, à savoir le *max-flow min-cut*. De plus, il est robuste aux pertes. En effet, la perte d'un paquet a moins d'impact sur les performances car l'information qu'il comporte peut être retrouvée dans d'autres paquets combinés. Ce type de codage permet aussi le passage à l'échelle car les combinaisons sont faites d'une manière décentralisée et utilisent des informations locales pour la collaboration entre les nœuds. Enfin, il permet une sécurité accrue car un paquet combiné n'est lisible qu'à condition d'avoir tous les

paquets utilisés pour créer la combinaison. Tous ces avantages font que le codage réseau s'applique dans plusieurs topologies de réseaux (cellulaires, ad hoc, etc.) et pour différentes applications (pair-à-pair, TCP, applications avec qualité de service, etc.).

L'utilisation du codage réseau dans un réseau de capteurs, dans le cas où il ne permet pas d'augmenter le débit, permet de simplifier la structure du réseau. Cette simplification vient du fait que toute forme de traitement est supprimée dans le réseau. Chacun des nœuds intermédiaires choisit les combinaisons uniformément et indépendamment des autres nœuds, ce qui permet de diminuer le coût de fabrication des nœuds de ce réseau. En contre-partie le codage réseau n'est pas sans inconvénient. En utilisant cette technique dans un réseau, on risque de perdre tous les paquets sources, à cause des erreurs. Des paquets corrompus intentionnellement peuvent être combinés avec d'autres paquets, ce qui entraine la perte totale des paquets source.

Dans ce chapitre, nous expliquons l'aspect théorique du codage réseau, son principe et l'intérêt de son utilisation, puis nous définissons le codage réseau linéaire. Dans la deuxième partie de ce chapitre nous définissons le codage réseau aléatoire ainsi que ses principales propriétés. Nous considérons le modèle d'un réseau à un seul saut et nous étudions ses performances en terme de probabilité de décodage. Ensuite, nous considérons le modèle d'un réseau multi-sauts. Nous prenons comme exemple d'application le scénario de collecte de données dans lequel des capteurs aléatoirement distribués envoient des données à une station de base (BS) en présence du bruit de fond et des paquets erronés injectés par un utilisateur malveillant. Nous proposons comme solution d'utiliser la concaténation des codes en métrique rang et des codes convolutifs pour faire face à ces type d'erreur. Dans la troisième partie de ce chapitre, nous présentons une comparaison, à travers des simulations, du modèle proposé par rapport à une solution présentée dans la littérature qui considère le même scénario en utilisant cette fois les codes en métrique rang concaténés avec les codes Reed-Solomon.

4.1 Le codage réseau

La technique du codage réseau a été introduite pour la première fois dans le but de maximiser le flux dans le réseau filaire. En effet, Ahlswede et al. ont introduit une nouvelle classe de problème qui s'appelle *le problème du flux d'information dans le réseau* que l'on peut voir comme une extension du théorème *max-flow min-cut* [PS98]. Ils se sont basés sur le fait que l'information que l'on envoie d'une source vers des destinations ne peut pas dépasser une certaine valeur, cette valeur s'appelle la capacité maximale du réseau ou le flot max. Le problème du max-flow min-cut consiste à trouver un flux qui maximise la capacité du réseau et qui soit réalisable depuis la source vers une destination. Ils ont montré que si on considère l'information comme une entité fixe que l'on peut seulement dupliquer ou transmettre, on ne peut pas atteindre la capacité maximale du réseau, contrairement au codage réseau qui permet d'atteindre cette valeur.

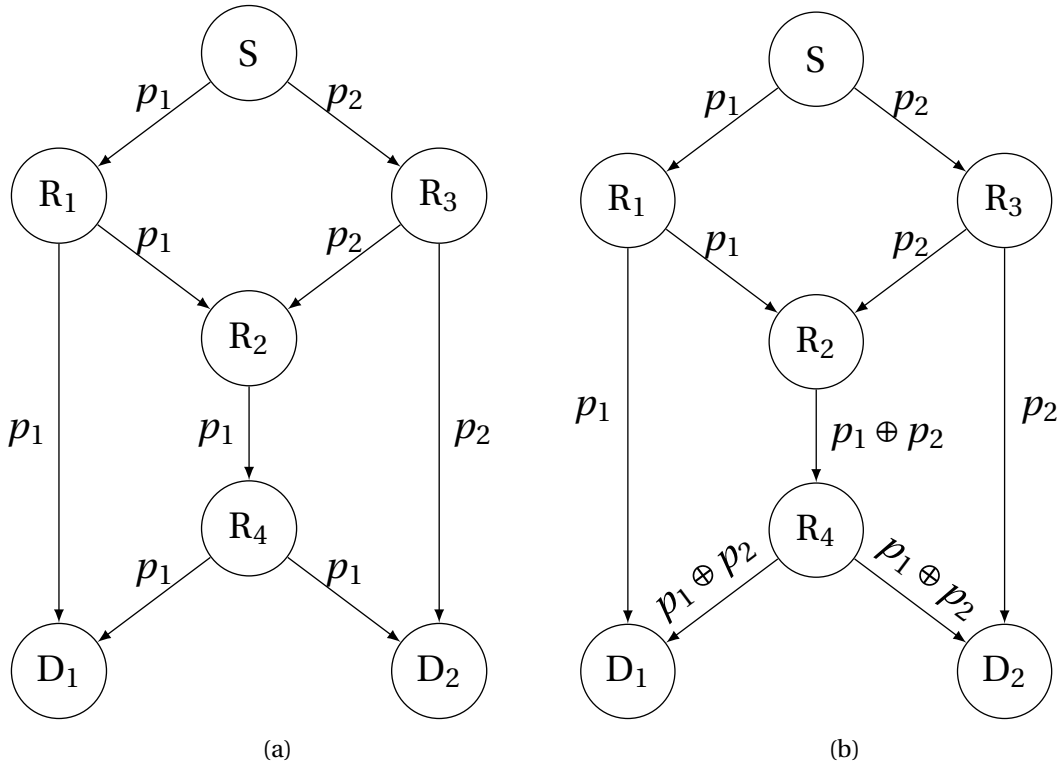


FIGURE 1 – Un réseau composé d'une source et deux destinations avec un flux mulitcast.

La Figure 1 illustre un exemple de réseau filaire composé d'une seule source, de deux destinations et de quatre relais. La source S génère deux paquets p_1 et p_2 pour les envoyer aux destinations D_1 et D_2. Supposons que la capacité de chaque ligne est égale à 1. La source S peut envoyer deux paquets à la destination D_1 à travers le chemin $S \rightarrow R_1 \rightarrow D_1$ et le chemin $S \rightarrow R_2 \rightarrow R_4 \rightarrow D_1$, de même pour la destination D_2. Le nombre de chemins qui relient la source à la destination D_2 est égale à 2. En appliquant le théorème du coupe min flot max la capacité maximale est deux paquets par créneau. Supposons maintenant que la source veut transmettre deux paquets aux deux destinations D_1 et D_2. La capacité maximale du réseau est le min des capacités pour chacune des destinations prises séparément. La capacité maximale du réseau est égale à deux dans cet exemple. Dans un scénario classique, store and forward, le nœud intermédiaire R_2 transmet le paquet p_1 puis p_2 au nœud R_4 si le paquet p_1 qui est reçu en premier. Ensuite, le nœud R_4 envoie le paquet p_2 à D_1 et le paquet p_1 à D_2. En revanche, dans le cas du codage réseau le nœud R_2 effectue un XOR des deux paquets reçus et envoie $p_1 \oplus p_2$ au nœud R_4. A son tour, le nœud R_4 transmet la combinaison à D_1 et D_2. Par conséquent, l'approche classique nécessite 4 transmissions pour que la destination D_1 reçoive le paquet p_1 et cinq transmissions pour que la destination D_2 reçoive le paquet p_2. Tandis que l'approche qui utilise la technique de codage réseau nécessite 4 transmissions dans tous les cas pour qu'un paquet atteigne sa destination. Donc, on peut gagner dans cet exemple jusqu'à 1/4 de temps de transmission. Cet approche a été généralisée dans le même article pour les réseaux multi-sources.

Pour bien illustrer cette idée, on considère un scénario simple décrit dans les fi-

gures 2 et 3 où deux sources S_1 et S_2 décident d'envoyer respectivement les paquets p_1 et p_2 en même temps l'un à l'autre à travers le relai R. La Figure 2 illustre un exemple de transmission en utilisant la technique du routage. Le relai R reçoit les deux paquets p_1 et p_2 et les renvoie vers les destinations. La Figure 3 illustre un exemple de transmission en utilisant la technique du codage, le relai R reçoit les paquets p_1 et p_2, effectue une combinaison des deux paquets et les envoie en broadcast aux deux destination. Le nombre de transmissions nécessaires pour que les deux destinations reçoivent les deux paquets est 3 transmissions dans le premier exemple et 2 transmissions dans le deuxième exemple. On déduit de cet exemple que l'utilisation du codage réseau permet de réduire le délai de la transmission dans le réseau.

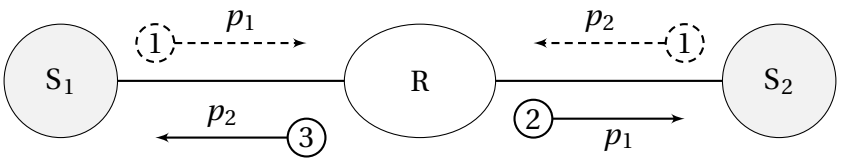


FIGURE 2 – Exemple de codage classique.

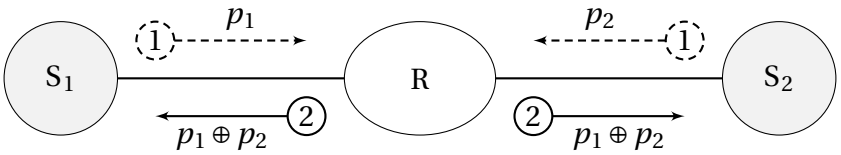


FIGURE 3 – Exemple du codage réseau.

4.1.1 Le codage réseau linéaire

Le problème de maximisation de la quantité d'information dans un réseau et la nécessité d'une solution pour ce problème ont mené à plusieurs travaux et nombreuses sont les propositions apportées dans ce domaine. Il s'agit en particulier de l'article [LYC03] dans lequel Li et al. ont étendu les travaux cités précédemment dans le cas des réseaux sans fil. Ils ont montré, que pour un trafic multicast, les codes linéaires multicast sont suffisants pour atteindre les bornes de la capacité maximale. Pour l'atteindre, il suffit de prendre la taille du corps considéré suffisamment grande pour les combinaisons linéaires. Une combinaison linéaire des vecteurs lignes $(\boldsymbol{p}_1, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_k)$ de longueur m dans un corps fini $\mathbb{F}_q$ est un élément l'espace engendré par ces vecteurs. Pour que deux combinaisons soient linéairement indépendantes avec une grande probabilité, il suffit de prendre q ou m suffisamment grand. Nous avons montré dans le chapitre 2 que la probabilité que des vecteurs soient linéairement indépendants dépend de q, de m, ainsi que du nombre de ces vecteurs.

Dans le codage réseau linéaire les paquets sont considérés comme des vecteurs de $\mathbb{F}_q$. Dans la pratique, on utilise des corps finis dont la taille q est égale à une puissance de 2. Ce choix permet d'avoir une implémentation efficace des opérations dans

le corps, en particulier, l'addition et la multiplication ce qui permet de résoudre facilement un système linéaire. A la sortie des nœuds intermédiaires, le paquet codé est de la forme :

$$\boldsymbol{p}' = \begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix} \times \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{km} \end{bmatrix}, \tag{4.1}$$

où $p_{i=1:k} = (\boldsymbol{p}_{i1}, \boldsymbol{p}_{i2}, \ldots, \boldsymbol{p}_{im})$ sont des vecteurs lignes à coefficient dans $\mathbb{F}_q$ et $\boldsymbol{a}$ correspond au vecteur de codage. Chaque paquet codé par le nœud intermédiaire ayant une liaison directe avec la source correspond à une combinaison des paquets source. En outre, les paquets codés par les autres nœuds intermédiaires qui ont reçu des paquets codés par d'autres nœuds sont également des combinaisons de paquets source. Ainsi, l'utilisation du codage réseau linéaire permet de simplifier l'architecture du réseau, et rend les opérations de codage et de décodage plus simple et moins complexe en terme d'implémentation dans la pratique.

Notons que lorsque le nœud intermédiaire ne reçoit pas de paquet, il n'effectue aucune opération et attend simplement le prochain créneau temporel. Dans le cas où le nœud dispose d'un seul paquet, il attend jusqu'à la fin de la durée de mise en tampon et envoie le paquet car il ne lui reste pas de temps pour faire les combinaisons. Dans un tel cas, le routage classique est plus efficace que le codage réseau puisqu'il envoie les paquets reçus dès leur réception. Dans le cas où le nœud intermédiaire reçoit plusieurs paquets simultanément, il peut effectuer une combinaison de tous les paquets. Dans ce cas, on risque de ne pas pouvoir décoder à la destination. Le nœud intermédiaire peut également effectuer plusieurs combinaisons, envoie une seule combinaison à chaque fois et garde les autres combinaisons dans la mémoire tampon. Dans le dernier cas, on risque de recevoir plus de paquets dans le créneau qui suit. Par conséquent, le choix de la durée de mise en tampon dépend essentiellement du nombre de paquets reçus. En effet, elle doit être choisi en fonction de nombre de paquet reçus pour éviter la perte des paquets.

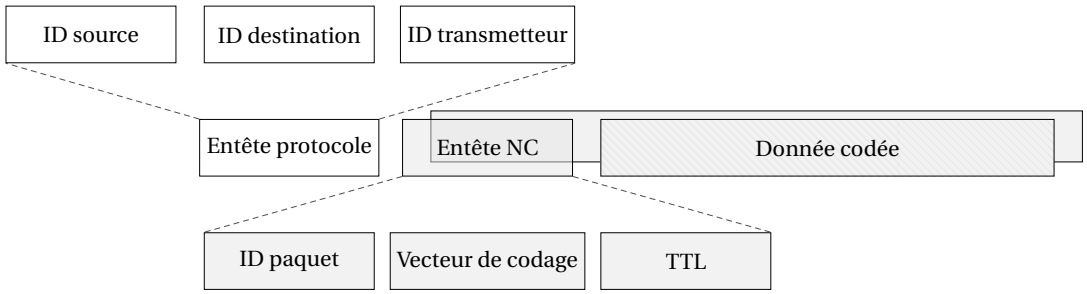


FIGURE 4 – La structure des messages codés du codage réseau.

Dans la pratique, chaque paquet généré après avoir appliqué la technique du codage réseau est composé principalement du message codé, de l'entête du codage ré-

seau ainsi que l'entête du protocole, comme on peut voir dans la Figure 4. L'entête du codage réseau se compose de l'identifiant de paquet, du vecteur de codage et de la durée de vie du paquet. L'entête du protocole se compose de l'identifiant de la source, de l'identifiant des destinations dans le cas multicast et de l'adresse du transmetteur qui a effectué la dernière combinaison. Cet entête peut contenir également d'autres informations telles que la route que doit suivre chaque paquet pour avoir un délai te transmission minimal dans le cas d'une gestion centralisée du réseau.

4.1.2 Approche algébrique

Des résultats similaires à ceux dans l'article [LYC03] sont trouvés par R. Kötter et M. Médard dans leur article [KM03], dans lequel ils ont introduit le codage réseau linéaire et ils ont proposé de résoudre le problème du flux d'information dans un réseau multicast dans un cadre algébrique. Ils ont montré que la capacité multicast du réseau peut être atteinte dans un réseau linéaire et ils ont donné les conditions pour lesquelles un ensemble de connexion soient réalisables dans un réseau.

L'approche algébrique est basée sur le fait que le réseau considéré est linéaire. Dans ce cas, la relation entre les paquets source et les paquets reçus par les destinations peut être exprimée par une matrice qu'on appelle *matrice de transfert* $\mathbf{M}$ à coefficient dans le corps de Galois $\mathbb{F}_q$. Le réseau peut être considéré comme un graphe direct. Dans ce cas, la matrice de transfert $\mathbf{M}$ est écrite sous la forme $\mathbf{M} = \mathbf{A}(\mathrm{I} - \mathbf{F})^{-1}\mathbf{B}_i^{\mathsf{T}}$, où $\mathbf{A}$ est la matrice de transfert de la source aux nœuds intermédiaires et $\mathbf{B}_i$ représente la matrice de transfert des nœuds intermédiaires à la destination i et $\mathbf{F}$ est la matrice d'adjacence obtenue à partir du graphe du réseau dans lequel chaque sommet représente un lien. Pour bien illustrer cette idée, on considère un exemple de réseau représenté dans la Figure 5. Pour plus de détail, le lecteur est invité à lire les articles [KM03] ; [Bas+13].

La source S désire envoyer deux paquets p_1 et p_2 aux destinations D_1 et D_2 à travers des nœuds intermédiaires.

Les nœuds R_1 et R_3 reçoivent respectivement les paquets $p'(e_1)$ et $p'(e_2)$ à travers les liens e_1 et e_2. Les paquets $p'(e_1)$ et $p'(e_2)$ vérifient

$$\begin{aligned} p'(e_1) &= a_{11}p_1 + a_{12}p_2 \\ p'(e_2) &= a_{21}p_1 + a_{22}p_2 \end{aligned}$$

Les paquets $p'(e_i)$ transmis par les liens e_i vérifient

$$\begin{aligned} p'(e_3) &= b_3 p'(e_1) \\ p'(e_4) &= b_4 p'(e_2) \\ p'(e_7) &= b_{5,1}p'(e_3) + b_{5,2}p'(e_4) \end{aligned}$$

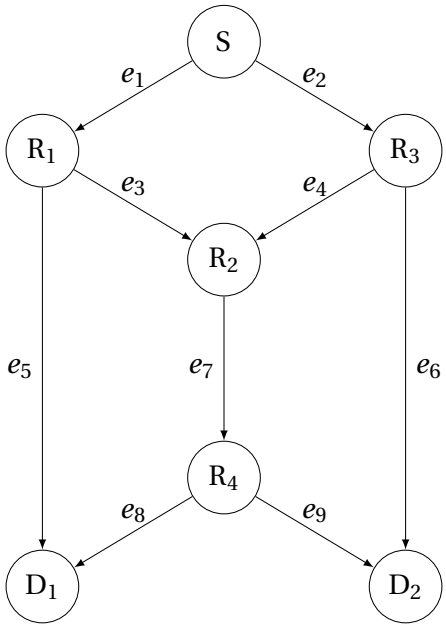


FIGURE 5 – Exemple de transmission de deux paquets par la source vers deux destinations via des relais [Bas+13].

Par la suite, les destinations D_1 et D_2 reçoivent quatre paquets vérifiant

$$\begin{aligned} p'(e_5) &= b_7 p'(e_1) \\ p'(e_6) &= b_6 p'(e_2) \\ p'(e_8) &= b_8 p'(e_7) \\ p'(e_9) &= b_9 p'(e_7) \end{aligned}$$

Par conséquent, les matrices de transfert M_1 et M_2, pour les destinations D_1 et D_2, vérifient

$$\begin{aligned} \mathbf{M}_1 &= \mathbf{A}(\mathrm{I}-\mathrm{F})^{-1}\mathbf{B}_1^{\mathsf{T}} \\ \mathbf{M}_2 &= \mathbf{A}(\mathrm{I}-\mathrm{F})^{-1}\mathbf{B}_2^{\mathsf{T}} \end{aligned}$$

avec

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

et F la matrice d'adjacence, peut être calculée à partir du graphe du réseau, illustré dans la Figure 6.

Le théorème du *réseau linéaire multicast*(Théorème 2 [KM03]) montre que, dans un réseau composé d'une source et plusieurs destinations, on peut atteindre la capacité multicast si et seulement si le flot-max est atteint pour chaque destination en utilisant le codage réseau linéaire. La destination n'a d'information sur la matrice de transfert que celle qui la concerne. Cependant, les auteurs de l'article ne donnent pas

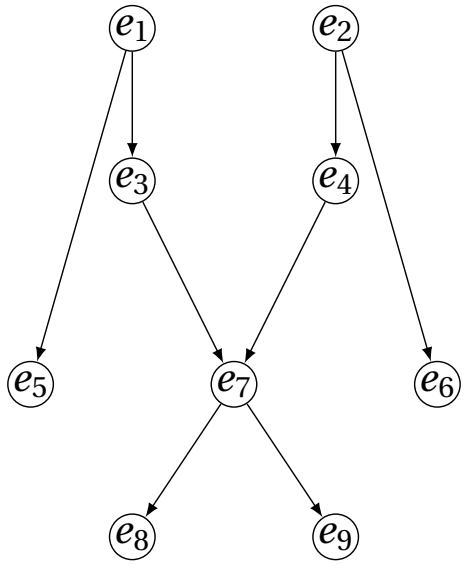


FIGURE 6 – Le graphe correspondant au réseau considéré dans la Figure 5 dans lequel chaque sommet représente un lien entre deux nœuds intermédiaire [Bas+13].

un algorithme efficace pour avoir les combinaisons effectuées par les nœuds intermédiaires, ce qui nécessite une connaissance centralisée de la topologie du réseau ainsi que des coefficients de codage. Ces contraintes rendent l'application du codage réseau dans la pratique très difficile.

4.1.3 Exemple de décodage du codage réseau

Pour mieux comprendre le processus de décodage du codage réseau, nous considèrerons un scénario où une source S veut envoyer des paquets vers une destination D. Ce scénario est semblable à celui présenté dans la partie précédente mis à part l'absence des relais dans ce scénario. Cela simplifie énormément l'architecture du réseau et dans ce cas la source effectue le codage réseau linéaire pour coder les k paquets générés et les envoie directement à la destination.

Nous présentons un exemple simple dans lequel la source dispose d'un message m à transmettre vers une destination à travers un canal sans erreur. Le message est fragmenté pour générer trois nouveaux messages p_1, p_2 et p_3 de même taille. Si la taille du troisième paquet n'est la même que celle du premier ou du deuxième on ajoute des zéros pour avoir la même taille pour les trois messages. La source code les trois messages en utilisant le codage réseau pour générer cinq paquets p'_1, p'_2, p'_3, p'_4 et p'_5. Ensuite elle les envoie à la destination, ce processus de codage est exprimé dans la Figure 7.

Après la réception des paquets source par la destination, cette dernière procède au décodage. Au début, la destination reçoit seulement le paquet p'_1 qui est la somme de deux paquets p_1 et p_2. Dans ce cas elle doit attendre la réception d'autres paquets pour pouvoir récupérer des informations dans ce paquets, comme on peut voir dans la Figure 8.

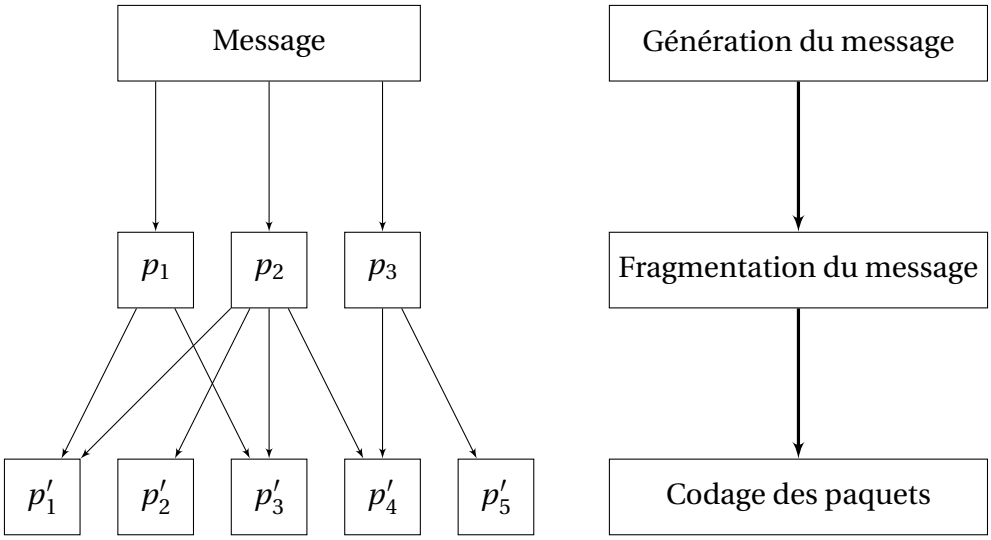


FIGURE 7 – Exemple du codage dans la source.

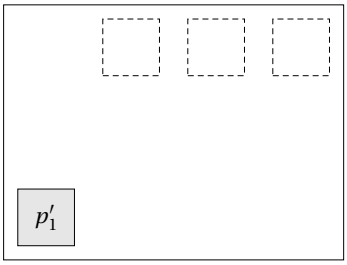


FIGURE 8 – L'état du décodeur à la destination après la réception du premier paquet.

Ensuite, la destination reçoit le deuxième paquet $p'_2 = p_2$ qui n'est combiné avec aucun autre paquet. Ce paquet est utilisé pour récupérer le paquet p_1 du paquet p'_1. Les paquets p'_1 et p'_2 seront supprimés par la suite. Cet étape du processus du décodage est représentée dans la Figure 9.

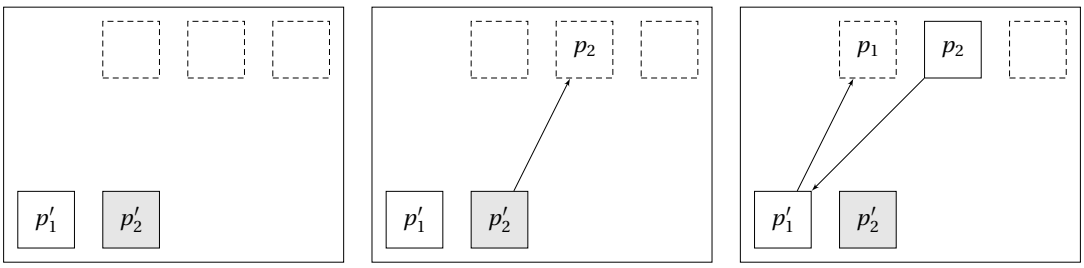


FIGURE 9 – L'état du décodeur à la destination après la réception de deux paquets.

Dans la troisième étape, la destination reçoit le paquet p'_3 contenant des paquets déjà récupérés Ce paquet n'est pas utile dans le processus de décodage, il sera donc rejeté (Figure 10).

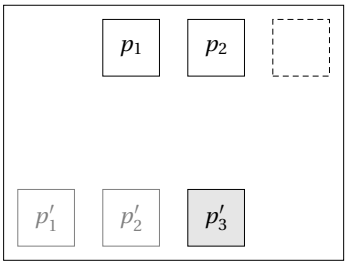


FIGURE 10 – L'état du décodeur à la destination après la réception de trois paquets.

La destination reçoit ensuite le quatrième paquet p'_4 qui est la somme du paquets p_2 et p_3. Puisqu'on dispose du paquet p_2, on peut directement récupérer le paquet p_3, comme on peut voir dans la Figure 11.

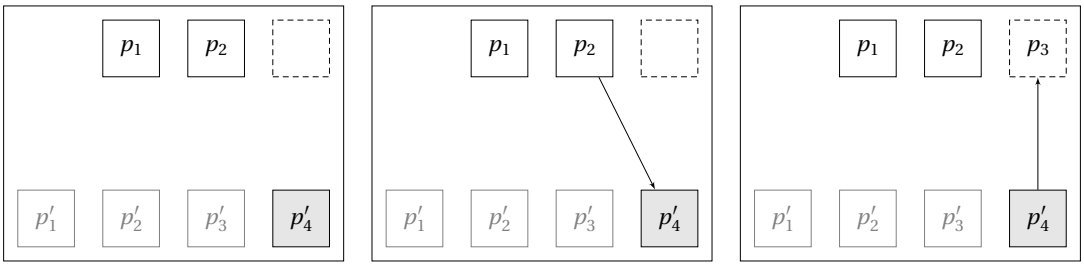


FIGURE 11 – L'état du décodeur à la destination après la réception de quatre paquets.

En général, le paquet récupéré est supprimé de tous les paquets reçus contenant ce paquet. Après avoir récupéré le dernier paquet, l'algorithme de décodage n'a pas besoin de recevoir le cinquième paquet et il s'arrête. Dans cet exemple, le processus de décodage s'est terminé avec succès. En effet, la destination n'a besoin que de 4 paquets pour récupérer les paquets envoyés par la source. Pourtant, le rendement du code dans cet exemple est $\frac{3}{5}$ puisque la source a envoyé 5 paquets. Ces paquets supplémentaires serviront à augmenter la probabilité de décodage dans le cas d'un canal à effacement.

4.2 Le codage réseau linéaire aléatoire

Pour faire face aux problèmes liés à l'utilisation du codage réseau cités auparavant, le codage réseau linéaire a été proposé par Ho, Kötter et Médard dans [Ho+03], et a été étendu dans [Ho+06] au codage réseau linéaire aléatoire. Le codage réseau linéaire aléatoire est une technique efficace pour la transmission de données dans un réseau non cohérent. Les auteurs ont proposé *le codage réseau linéaire aléatoire*, RLNC, une approche très différente de ce que nous avons présenté dans la section précédente. RLNC consiste à choisir les combinaisons au sein des nœuds intermédiaires d'une manière aléatoire dans un corps fini $\mathbb{F}_q$ de taille suffisamment large. En effet, Ho, Kötter et Médard ont montré que si la taille du corps est suffisamment large, la capacité du réseau est atteinte avec une probabilité proche de 1 si les nœuds intermédiaires choisissent d'une manière aléatoire les combinaisons des paquets reçus. Cela permet d'augmenter le débit ainsi que de transmettre les données d'une manière distribuée,

optimale et efficace contre les pertes. Si on a supposé que pour atteindre la capacité du réseau, il faut une autorité centralisée qui décide le chemin que doivent prendre les paquets pour arriver à une destination, ainsi que les opérations du codage que les nœuds intermédiaires doivent effectuer. L'importance de l'approche du RLNC réside dans l'absence d'une autorité centralisée, ce qui simplifie l'architecture du réseau. Pour cette raison, l'application du codage réseau dans un réseau réel est devenue possible. On peut citer comme domaine d'application les réseaux P2P filaires, les réseaux de capteurs et les réseaux cellulaires.

Le codage réseau linéaire aléatoire est une théchnique efficace de transmission de données dans un réseau non cohérent [Ho+03]. Les paquets codés sont considérés comme des vecteurs de longueurs N sur le corps $\mathbb{F}_q$. Dans ce cas, les nœuds intermédiaires choisissent, d'une manière aléatoire, les combinaisons pour coder les paquets envoyés par la source. Bien que le codage réseau aléatoire ne garantisse pas dans tous les cas que tous les paquets soient correctement transmis dans un délai court, des travaux montrent que pour une taille de corps fini assez large, la probabilité de pouvoir récupérer les paquets envoyés par la source en un minimum de transmission tend exponentiellement vers 1. En revanche, la complexité de calcul par les nœuds intermédiaires et la complexité de décodage dépend du choix de la taille du corps $\mathbb{F}_q$. C'est pour cela que des corps du type $\mathbb{F}_{2^8}$ ou $\mathbb{F}_{2^{16}}$ sont proposés dans la plupart des applications.

4.2.1 L'approche générale

Considérons un scénario simple de réseau comportant une source S, un nombre aléatoire de relais et une destination D. La source S désire envoyer des données à la destination D. Les données à envoyer sont divisées en k blocs de longueur m dans le corps de Galois $\mathbb{F}_q$. Un identifiant est ajouté à chaque bloc pour remettre en ordre les paquets lors de leur réception par la destination. Les blocs générés sont considérés comme des vecteurs de l'espace vectoriel $\mathbb{F}_q^m$. Ensuite, ils sont combinés aléatoirement entre eux, en utilisant l'équation (4.1), pour former N nouveaux blocs. Cela signifie qu'on introduit de la redondance afin d'avoir un nombre de paquets suffisant pour le décodage. La taille du corps $\mathbb{F}_q$ est choisie de telle façon à ce que les paquets générés soient linéairement indépendants avec une probabilité proche de 1. Le choix du nombre des paquets générés N dépend du canal (s'il y a des effacements ou non) ainsi que de la taille du corps choisi.

Les N paquets générés sont de la forme :

$$\begin{bmatrix} p'_1 \\ p'_2 \\ p'_3 \\ \vdots \\ p'_{\mathrm{N}} \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1k} \\ g_{21} & g_{22} & \cdots & g_{2k} \\ g_{31} & g_{32} & \cdots & g_{3k} \\ \vdots & \ddots & \ddots & \vdots \\ g_{\mathrm{N}1} & g_{\mathrm{N}2} & \cdots & g_{\mathrm{N}k} \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix}, \tag{4.2}$$

tel que la matrice G est la matrice du codage où chaque ligne de cette matrice représente le vecteur codage utilisé pour générer les paquets à transmettre. Cette étape

est cruciale pour éviter les pertes des paquets dans le réseau que ce soit à cause des effacements ou des combinaisons effectué au seins des nœuds intermédiaires qui peuvent à la fin générer des combinaisons linéairement dépendantes.

Après la génération des paquets codés, la source envoie ces derniers aux nœuds intermédiaires. A son tour, chaque nœud intermédiaire effectue un codage réseau aléatoire sur les paquets reçus. Ce processus est répété jusqu'à ce que la destination reçoive les paquets sources. Les combinaisons effectuées par les nœuds intermédiaires sont des combinaisons linaires. Cela implique que les paquets reçus par la destination sont des combinaisons des paquets source. La destination reçoit des combinaisons de la forme :

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{\mathrm{N}'} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ a_{31} & a_{32} & \cdots & a_{3k} \\ \vdots & \ddots & \ddots & \vdots \\ a_{\mathrm{N}'1} & a_{\mathrm{N}'2} & \cdots & a_{\mathrm{N}'k} \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix}. \tag{4.3}$$

La matrice transfert A est un produit de la matrice de génération et de la matrice de combinaison. Le décodage est possible si la matrice A est de rang égale à k. Nous avons donné dans le chapitre 2 la probabilité pour qu'une matrice générée de façon aléatoire soit inversible.

4.2.2 Réseau à un seul saut

Nous analysons, dans cette partie, le codage réseau aléatoire dans un réseau à un seul saut. Dans ce cas le codage réseau est considéré comme un code correcteurs d'erreurs dans un canal à effacement i.e., tout paquet reçu est considéré bien reçu et conservé ou corrompu et rejeté. Un exemple similaire des codes à effacement est la famille des codes fontaine comme les LT codes (Luby Transform) [Lub02] et les codes Raptor [Lub+06]. Le principe du codage dans un canal à effacement est de pouvoir récupérer le message initial sans avoir besoin de l'information perdue lors de la transmission. L'utilisation d'un code correcteur classique, comme les codes Reed Solomon (RS) ou les Codes Low Density Parity Check (LDPC), a des limitations dans le cas d'un canal à effacement. Dans le cas des réseaux multicast, le rendement du code utilisé est adapté à la situation des nœuds destination. En général, il est choisit de telle sorte que tous les destinations reçoivent les paquets source. De ce fait, les destinations qui ont un bon canal sont obligées d'attendre la réception de tous les paquets. De plus, si le nombre de paquets perdus est supérieure à la capacité de détection du code, les données reçues ne sont pas utilisable par la destination et donc la source doit retransmettre tous les paquets depuis le début. En revanche, les problèmes que nous venons de citer ne se posent pas lors de l'utilisation de codage réseau aléatoire. En effet, dans le cas des réseaux multicast, chaque récepteur a besoin d'un nombre de paquets source N_i pour pouvoir décoder en fonction du canal. Le nombre de paquets transmises par la source N égal au maximum des N_i. Après avoir reçu le nombre de paquets nécessaires pour le décodage, les destinations commencent le processus de décodage et se mettent en veille.

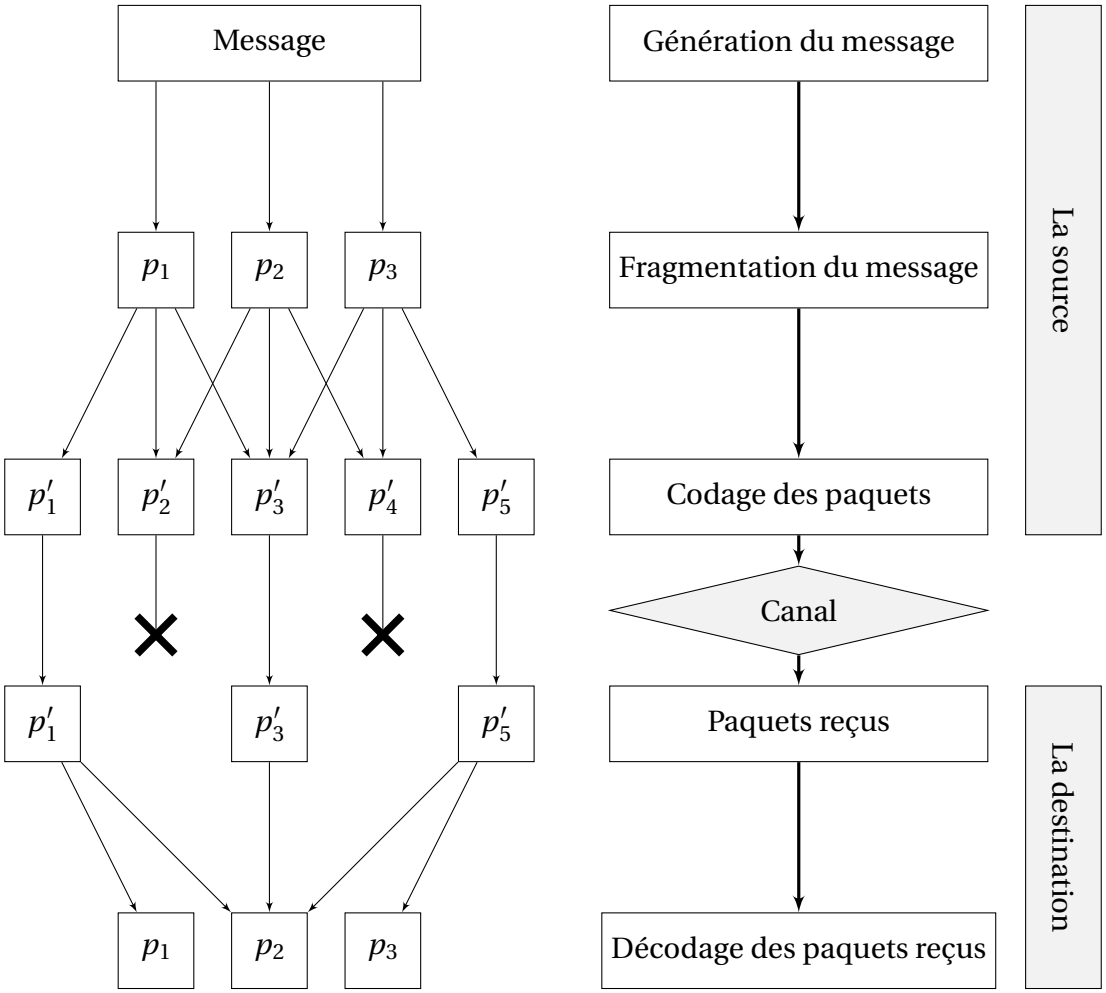


FIGURE 12 – Phase de codage/décodage en utilisant le codage réseau aléatoire.

Exemple de décodage

Nous considérons un scénario où une source décide d'envoyer k paquets vers une destination. Ce scénario est une extension de l'exemple présenté auparavant. Par contre dans cet exemple l'envoi des paquets se fait à travers un canal à effacement. Dans ce cas, la source effectue le codage réseau aléatoire, en utilisant l'équation (4.2), pour coder k paquets générés et les envoie directement à la destination.

Dans l'exemple illustré dans la Figure 12, la source génère trois paquets et les code en utilisant une matrice de transfert de taille 5×3. Ensuite, elle transmet les paquets codés à la destination à travers un canal à effacement. La destination reçoit trois paquets de la source et les deux autres sont perdus durant la transmission à cause de la force du signal. Ces paquets peuvent aussi être perdu par la destination à cause de la présence des erreurs dans les deux paquets ce qui implique le rejet de ces deux derniers. Pourtant, la destination possède trois paquets source. Donc, sans prendre en compte les deux paquets perdus, si la matrice de transfert est de rang plein on peut récupérer les paquets d'origine grâce à la redondance.

Dans la Figure 12, nous utilisons le corps de Galois $\mathbb{F}_5$. Les paquets sont générés à partir de la matrice de transfert A vérifiant

$$A = \begin{pmatrix} 3 & 0 & 0 \\ 1 & 4 & 0 \\ 1 & 2 & 3 \\ 0 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

La matrice de transfert reçue par la destination ne contient pas la deuxième et la quatrième lignes. Par contre, la destination arrive à récupérer les paquets envoyés par la source car les lignes de la matrice de transfert forment des vecteurs linéairement indépendants. Par conséquent, le codage réseau aléatoire permet d'augmenter le débit dans un réseau et de rendre la communication plus robuste aux effacements. La taille du corps utilisé doit être suffisamment grande pour garantir l'indépendance linéaire entre les combinaisons. Dans ce cas, d'autres paquets sont générés par les nœuds intermédiaires pour remplacer les paquets perdus. Comme nous avons vu dans l'exemple précédent, il n'est pas obligatoire de recevoir l'intégralité des paquets générés par la source grâce à la redondance. Dans le processus de décodage, il suffit que chaque destination reçoive une partie de la matrice de transfert, qui n'est pas forcement la même, pour commencer la phase de décodage. Le décodage consiste à résoudre un système linaire pour récupérer le message original.

Probabilité de décodage

Le nombre de paquets N, nécessaire pour qu'une source puisse récupérer les paquets, envoyés par la source est supposé connu dans les exemples traités auparavant. La connaissance de N est importante pour éviter l'utilisation du mécanisme traditionnel de la retransmission. Sans l'utilisation d'un *feedback*, la destination reçoit un nombre de paquets suffisant pour le décodage. Dans cette partie, nous donnons l'expression exacte de la probabilité de décodage après la réception de N paquets. Cette probabilité sera notée $\mathbb{P}_{\mathrm{R}}(\mathrm{N}, k)$, où k désigne le nombre de paquets générés par la source avant le codage.

Afin d'obtenir l'expression exacte de $\mathbb{P}_{\mathrm{R}}$, on se place dans un corps fini $\mathbb{F}_q$. Les paquets générés forment des combinaisons aléatoire des blocs du message initial. Cela implique que la matrice de codage est tiré aléatoirement dans l'espace $\mathbb{F}_q^{\mathrm{N}\times k}$. On distingue deux cas. Dans le premier cas, la matrice du codage est purement aléatoire. Alors, elle peut contenir des vecteurs lignes nuls. Dans le deuxième cas, la source ne génère dans aucun cas des combinaisons nulles. Dans le premier cas, la probabilité de recevoir k paquets linéairement indépendants lorsque $k \leq \mathrm{N}$ est :

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}, k, q\right) = \frac{\mathrm{S}(\mathrm{N}, k, q, k)}{q^{\mathrm{N}k}} = \prod_{i=0}^{k-1}\left(1 - q^{i-\mathrm{N}}\right), \tag{4.4}$$

où S peut être récupéré en utilisant l'équation(2.3) du chapitre 2. Dans le deuxième cas, le nombre de possibilité qu'on a pour tirer un vecteur non nul de $\mathbb{F}_q^k$ est $q^k - 1$. Dans

le deuxième tirage, il faut que le vecteur tiré soit non nul et linéairement indépendant avec le premier. Le nombre de possibilité est $q^k - q$, etc... Le nombre de possibilité de tirer aléatoirement un paquet qui soit linéairement indépendant avec j vecteurs qui sont linéairement indépendants entre eux est :

$$\mathbb{P}_{\mathrm{R}}^{j}\left(k, j, q\right) = \left(q^k - q^j\right). \tag{4.5}$$

On déduit la probabilité de décoder k paquets après la réception de k paquets, qui est la probabilité du tirage successive de premier vecteur sachant qu'il n'est pas nul, du deuxième vecteur sachant qu'il est linéairement indépendant avec le deuxième..., sur le nombre de possibilité d'avoir tous les vecteurs non nuls

$$\mathbb{P}_{\mathrm{R}}\left(k, k, q\right) = \prod_{i=1}^{k-1}\left(\frac{q^i - 1}{q^k - 1}\right). \tag{4.6}$$

Supposons que l'on tire $\mathrm{N} = k + 1$ vecteurs non nuls. Pour que k vecteurs soient linéairement indépendants, il faut que k vecteurs, parmi les $k+1$ vecteurs, soient linéairement indépendants et un seul vecteur est linéairement dépendant des k vecteurs. Ce vecteur peut être linéairement dépendant du premier vecteur tiré ou du premier et deuxième... On a

$$\mathbb{P}_{\mathrm{R}}\left(k+1, k, q\right) = \prod_{i=0}^{k-1}\left(\frac{q^i - 1}{q^k - 1}\right)\sum_{j=1}^{k}\left(\frac{q^j - 1}{q^k - 1}\right), \tag{4.7}$$

En suivant le même raisonnement, on obtient l'expression générale de la probabilité de décodage pour $k \leq \mathrm{N}$

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}, k, q\right) = \prod_{i=1}^{k-1}\left(\frac{q^i - 1}{q^k - 1}\right)\sum_{j_1=0}^{k}\left(\frac{q^{j_1} - 1}{q^k - 1}\right)\cdots\sum_{j_{\mathrm{N}-k}=j_{\mathrm{N}-k-1}}^{k}\left(\frac{q^{j_{\mathrm{N}-k}-1}}{q^k - 1}\right), \tag{4.8}$$

Cette formule peut être simplifiée en utilisant l'équation 11 dans [TCBOF11], l'équation (4.8) devient :

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}, k, q\right) = \mathbb{P}_{\mathrm{R}}^{0} \times \left(\begin{bmatrix}\mathrm{N}\\ \mathrm{N}-k\end{bmatrix}_q + \sum_{i=1}^{k}(-1)^i\binom{\mathrm{N}}{i}\begin{bmatrix}\mathrm{N}-i\\ \mathrm{N}-k-i\end{bmatrix}_q\right), \tag{4.9}$$

où $\mathbb{P}_{\mathrm{R}}^{0}$ vérifie :

$$\mathbb{P}_{\mathrm{R}}^{0} = \frac{q^{k^2}}{\left(q^k - 1\right)^{\mathrm{N}}}\prod_{i=1}^{k}\left(1 - \frac{1}{q^i}\right)$$

La probabilité de décodage, exprimée dans l'équation (4.9), peut être simplifiée grâce à l'approximation des termes binomiaux. Dans son article [Zha12], Zhao montre que la probabilité de décodage est exprimée comme suit :

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}, k, q\right) = \frac{\sum_{i=0}^{\mathrm{N}-k}(-1)^i\binom{\mathrm{N}}{i}\prod_{j=0}^{k-1}\left(q^{\mathrm{N}-i} - q^j\right)}{\left(q^k - 1\right)^{\mathrm{N}}}. \tag{4.10}$$

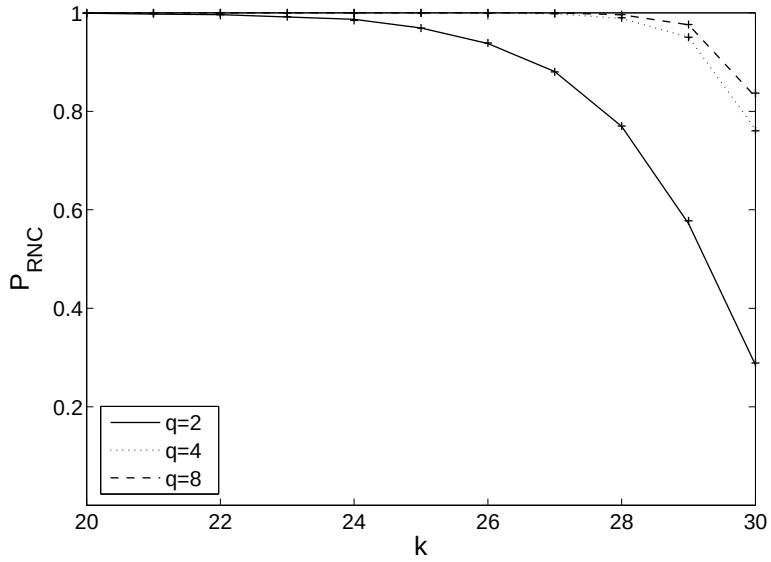


FIGURE 13 – Probabilité de décodage en fonction de k après réception de N = 30 paquets codés.

L'expression théorique est validée par des simulations pour différentes valeurs de q dans la Figure 13. Nous pouvons remarquer que la taille du corps q a un grand impact sur la probabilité de décodage. Cela prouve ce que nous avons dit dans l'introduction de ce chapitre à propos du choix de q. Nous avons simulé la probabilité $\mathbb{P}_{\mathrm{R}}$ en fonction de k, pour N = 30. Donc, $\mathbb{P}_{\mathrm{R}}$ est la probabilité que la destination arrive à décoder les N paquets reçus pour récupérer les k paquets générés par la source. Il est évident que, si la destination reçoit un nombre de paquets N inférieur à k, on ne peut pas commencer le processus de décodage. Donc, on doit s'assurer que l'on a reçu un nombre de paquets suffisant pour commencer la phase de décodage.

4.2.3 Réseau multi-sauts

Nous considèrerons un scénario où une source décide d'envoyer k paquets vers une destination à travers des relais. La source commence par ajouter à chaque paquet un vecteur. Chaque vecteur ajouté contient 1 dans une position et tous les autres positions sont nuls. Deux paquets ne peuvent pas avoir deux coefficients non nuls dans la même position. Ce choix est fait pour repérer les paquets après les avoir combiné. Les message générés sont de la forme

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix} = \begin{bmatrix} 1 & & & & m_1 \\ & 1 & & & m_2 \\ & & \ddots & & \vdots \\ & & & 1 & m_k \end{bmatrix}$$

Après avoir appliqué le lifting, la source utilise le codage réseau aléatoire, en appliquant l'équation (4.2), pour générer N paquets. Les paquets codés sont transmis par la suite vers la destination à travers des relais. Les relais reçoivent les paquets envoyés par la source ou par d'autres relais. Chaque relai applique une ou plusieurs

combinaisons et les envoie aux autres relais jusqu'à ce que les paquets arrivent à la destination. Puisque les paquets reçus par la destination sont des combinaisons linéaires des paquets générés par la source, comme il est décrit dans l'équation (4.3), l'espace généré par ces paquets est inclus ou égal à l'espace généré par les paquets source. Les deux espaces sont égaux avec une grande probabilité pour une taille de corps suffisamment grande. Par conséquent, le codage réseau peut être considéré comme la transmission d'un espace vectoriel de dimension constante.

Notons que la génération d'un grand nombre de paquets par la source augmente le risque de perte de paquets ainsi que la complexité de décodage. En effet, si la source génère un grand nombre de paquets, elle risque de s'approcher ou de dépasser la capacité du réseau. On risque, dans ce cas, de perdre la totalité de paquets ou d'envoyer des combinaisons supplémentaires pour pouvoir décoder. La complexité du décodage dépend de plusieurs paramètres y compris le nombre de paquets générés par la source.

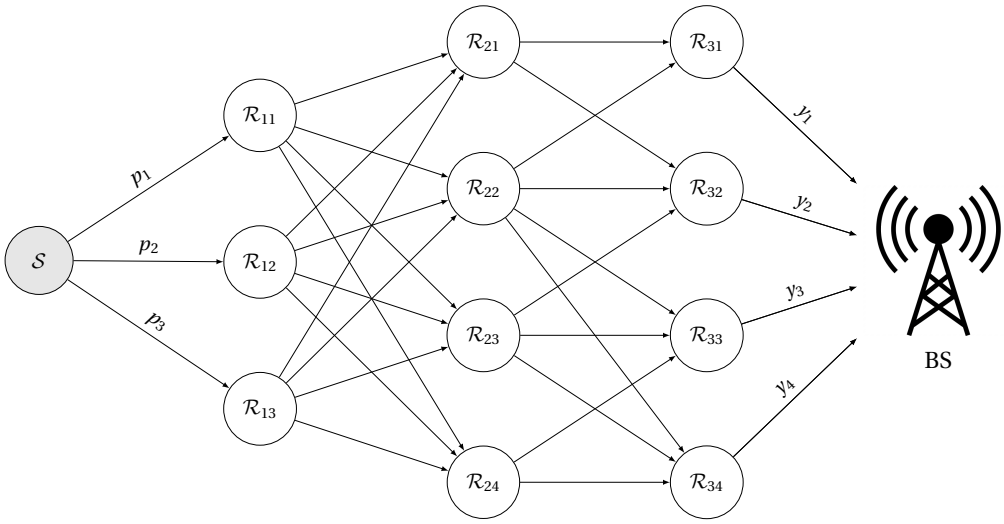


FIGURE 14 – Modèle de transmission utilisant le codage réseau aléatoire dans un réseau de capteurs comprenant une source, une station de base et des relais.

Considérons l'exemple illustré dans la Figure 14 d'un réseau de capteurs comprenant une source, onze nœuds intermédiaires et une station de base (BS). La source génère trois paquets m_1,m_2 et m_3. En suite, elle applique le lifting aux messages générés. La source applique le codage réseau pour coder les paquets générés. Vu que le rendement dans ce cas est égal à 1, la source choisit la matrice identité comme matrice de codage et elle transmet les paquet générés p_1, p_2, p_3 vers la destination. Supposons que la capacité de chaque ligne est égale à 1. La source S peut donc envoyer les trois paquets à la destination. La capacité de réseau dans cette exemple est égale à 3. Cela implique que le rang de la matrice de transfert A est inférieur ou égal à 3. La destination reçoit 4 paquets p'_1, p'_2, p'_3 et p'_4 et procède au décodage. La destination récupère les paquets codés par la source avec une probabilité $\mathcal{P}_{\mathrm{R}}\left(4,3,q\right)$ exprimée dans l'équation (4.10).

Soit N' le nombre de paquets reçus par la destination et k le nombre de paquets générés par la source. Supposons que k est inférieur à la capacité du réseau. Si ce n'est pas le cas, on prend k égale à la capacité du réseau. Nous exprimons maintenant la probabilité $\mathcal{P}_{\mathrm{R}}$ pour que le rang de la matrice de transfert soit égal à n en fonction de N, k, n et q.

Proposition 4.1. *La probabilité* $\mathcal{P}_{\mathrm{R}}\left(\mathrm{N}', k, n, q\right)$ *vérifie*

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}', k, n, q\right) = \begin{bmatrix} k \\ n \end{bmatrix}_q \frac{\sum_{i=0}^{\mathrm{N}'-n} (-1)^i \binom{\mathrm{N}'}{i} \prod_{j=0}^{n-1} \left(q^{\mathrm{N}'-i} - q^j\right)}{\left(q^k - 1\right)^{\mathrm{N}'}}. \tag{4.11}$$

Démonstration. En utilisant l'équation (4.10), le nombre de possibilités de générer n paquets linéairement indépendants sachant qu'on a reçu N' paquets est égal à

$$\sum_{i=0}^{\mathrm{N}'-n} (-1)^i \binom{\mathrm{N}'}{i} \prod_{j=0}^{n-1} \left(q^{\mathrm{N}'-i} - q^j\right).$$

D'après l'équation (2.2), le nombre de sous espace vectoriels de $\mathbb{F}_q^k$ de dimension n sur $\mathbb{F}_q$ est égal à $\begin{bmatrix} k \\ n \end{bmatrix}_q$.
Le nombre total de possibilités pour que la matrice de transfert soit de rang n est égal à

$$\begin{bmatrix} k \\ n \end{bmatrix}_q \sum_{i=0}^{\mathrm{N}'-n} (-1)^i \binom{\mathrm{N}'}{i} \prod_{j=0}^{n-1} \left(q^{\mathrm{N}'-i} - q^j\right).$$

Enfin, le nombre total de possibilités pour avoir une matrice de N-lignes et k-colonnes sachant que tous les vecteurs lignes sont non nuls est égal à $\left(q^k - 1\right)^{\mathrm{N}'}$. D'où le résultat. □

Ce résultat est la généralisation de l'équation (4.10). En effet, pour $n = k$, le binôme vérifie $\begin{bmatrix} k \\ k \end{bmatrix}_q = 1$. Donc, l'équation 4.11 devient

$$\mathbb{P}_{\mathrm{R}}\left(\mathrm{N}', k, k, q\right) = \frac{\sum_{i=0}^{\mathrm{N}'-k} (-1)^i \binom{\mathrm{N}'}{i} \prod_{j=0}^{k-1} \left(q^{\mathrm{N}'-i} - q^j\right)}{\left(q^k - 1\right)^{\mathrm{N}'}}.$$

4.3 Modèle de collecte de données en présence d'erreurs

Dans cette partie, nous présentons le modèle utilisé pour les simulations du comportement des codes correcteurs en métrique rang dans un réseau utilisant le codage réseau linéaire aléatoire. Nous considérons le modèle de transmission, illustré dans la Figure 15, d'une source qui désire envoyer k paquets à la station base.

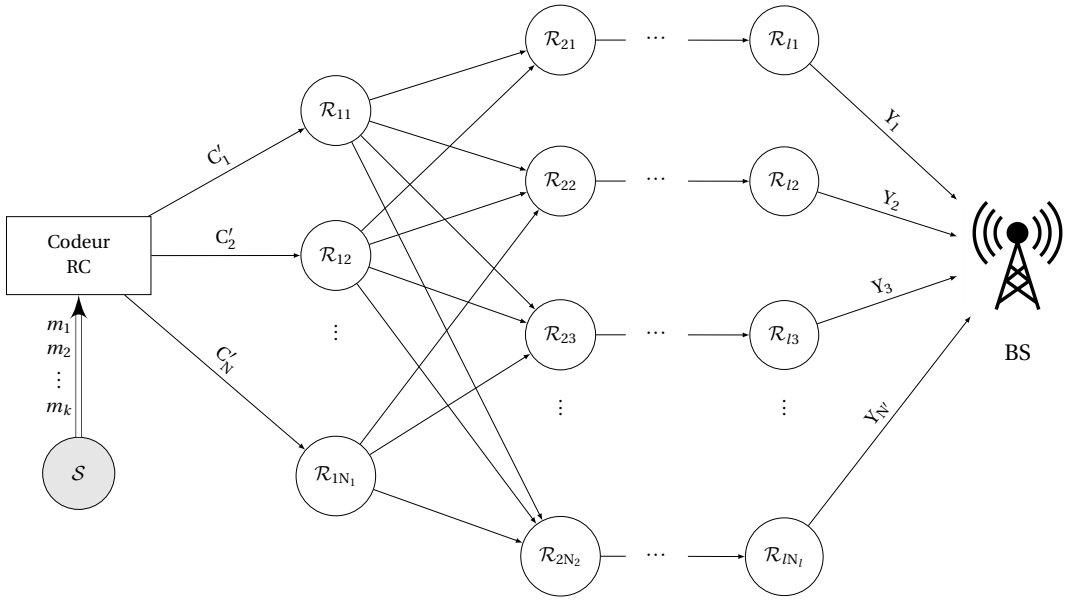


FIGURE 15 – Modèle de transmission dans un réseau de capteurs sans fils utilisant le codage réseau aléatoire et les codes en métrique rang.

Les capteurs sont généralement alimentés par des piles, et donc, nécessitent une faible consommation d'énergie pour maximiser leur durée de vie. Avec une gamme de transmission limitée, la collecte de données nécessite l'utilisation des relais pour atteindre la station de base. La source commence par coder les données collectées en utilisant un code LRPC de paramètres $[\mathrm{N}, k, d]_{q^m}$ dans $\mathbb{F}_{q^m}$. Ensuite, elle applique le lifting sur les paquets en sortie du codeur

$$\begin{bmatrix} \mathrm{C}'_1 \\ \mathrm{C}'_2 \\ \vdots \\ \mathrm{C}'_k \end{bmatrix} = \begin{bmatrix} 1 & & & \mathrm{C}_1 \\ & 1 & & \mathrm{C}_2 \\ & & \ddots & \vdots \\ & & 1 & \mathrm{C}_k \end{bmatrix}$$

Le vecteur C_i est la i-ème colonne de la matrice C, où C est la transformation du mot de code c sous une forme matricielle. Dans le cas où on utilise les codes LRPC matriciels, le mot en sortie du codeur a systématiquement une forme matricielle.

Les paquets codés sont transmis par la suite vers les nœuds intermédiaires. Ces derniers appliquent le codage réseau aléatoire et envoient les paquets codés vers la BS. La transmission d'un niveau à un autre peut être considérée comme un modèle de transmission à un seul saut de plusieurs sources vers plusieurs destinations.

La station de base attend la réception d'un nombre de paquets suffisant pour le décodage et récupère les paquets codés en utilisant la méthode de résolution des systèmes linéaires. Ensuite, elle applique l'algorithme de décodage des codes en métrique rang.

4.3.1 Formulation du problème

Les paquets reçus par la destination $Y_1, Y_1, \ldots, Y_N$ sont des combinaisons linéaires des paquets $C'_1, C'_1, \ldots, C'_N$ envoyés par la source. Alors, on a

$$Y = A \times C' + E, \tag{4.12}$$

où A est la matrice de transfert de taille $N' \times N$ et E est la matrice des erreurs générées au cours de la transmission.

La matrice de transfert peut être vue comme la multiplication des matrices de transfert d'un niveau à un autre. Donc, A peut s'écrire sous la forme $A = A_1 \times A_2 \times \cdots \times A_l$, où A_i désigne la matrice de transfert entre le niveau $l-1$ et le niveau l. Puisque la source n'a pas effectué le codage réseau, la matrice A_1 est égale à la matrice identité. Notons que la taille de la matrice de transfert peut varier d'un niveau à l'autre mais elle n'a pas d'influence sur la probabilité de décodage tant que le nombre de lignes des matrices de transfert est supérieur à la capacité du réseau.

On peut distinguer deux types d'erreurs qui peuvent se produire dans un réseau de capteurs, les erreurs rang et les erreurs de propagation. On a

$$E = E^{(r)} + E^{(n)},$$

où $E^{(r)}$ est l'erreur rang et $E^{(n)}$ est l'erreur de propagation. On peut imaginer deux scénarios dans lesquels les erreurs de type rang peuvent se présenter lors de la transmission. Le premier scénario est lorsqu'un utilisateur malveillant injecte des paquets erronés dans le réseau pour perturber le système de transmission. Ce scénario est possible du moment que les nœuds intermédiaires reçoivent un grand nombre de paquets en provenance de la source ou des autres nœuds. Ces capteurs ne sont pas bien équipés pour la vérification de l'authenticité et de l'intégrité de tous les paquets reçus. Le deuxième scénario est représenté par la défaillance des nœuds intermédiaires ce qui conduit à la diminution de la capacité de réseau. Il se peut dans ce cas que la destination ne reçoive pas suffisamment de paquets pour procéder au décodage. Ce scénario conduit à la présence d'effacements plutôt que d'erreurs de type rang. Dans les deux cas, l'erreur rang peut s'écrire sous la forme $E^{(r)} = D^{(r)}Z^{(r)}$ où r est le rang de la matrice $E^{(r)}$, $D^{(r)}$ est une matrice de taille $N' \times r$ et $Z^{(r)}$ est une matrice de taille $r \times (N+m)$.

Les erreurs de propagation sont causées par la nature du canal sans fil. Ce type d'erreurs se caractérise par le fait que les positions erronés d'un paquet forme une distribution aléatoire. Les erreurs sont éparpillées dans dans la matrice des paquets de chaque niveau ce qui implique que cette matrice soit de rang plein.

Notons que lors de la multiplication de $\mathbf{A}_i$, dans chaque niveau, par la matrice de canal, les erreurs lignes changent et se transforment en erreurs colonnes comme nous pouvons le voir dans la Figure 16. Ensuite, le produit de $\mathbf{A}_i$ et $\mathbf{E}_i^{(n)}$ est ajouté à l'erreur de canal du niveau $i+1$. Cette opération est répétée dans chaque niveau du réseau jusqu'à l'arrivée des paquets à la station de base. Par ailleurs, dans le cas où le nombre

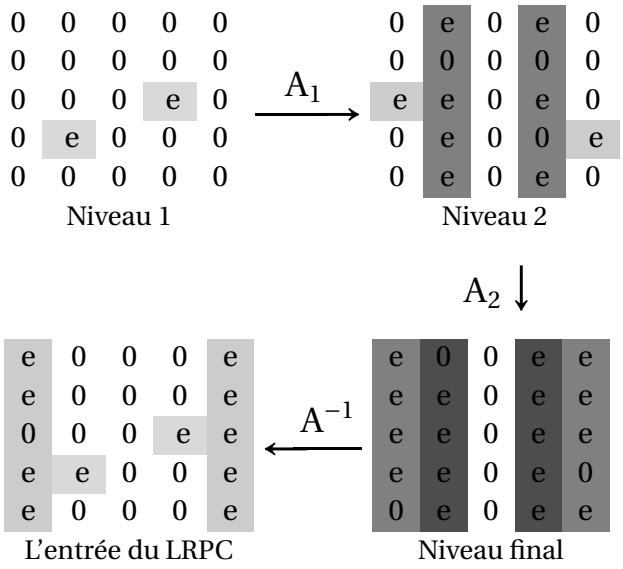


FIGURE 16 – Exemple de transmission de paquets dans un réseau RLNC en présence de bruit de fond.

de niveaux est important, la matrice d'erreur de propagation $\mathbf{E}^{(n)}$ est de rang plein ou presque. Dans ce cas, le code en métrique rang ne peut pas décoder les paquets reçus.

Dans la suite de cette partie, nous montrons l'impact des erreurs de propagation lors d'une transmission des paquets par une source utilisant les codes en métrique rang dans un réseau RLNC. Considérons le cas où la matrice de transfert est de rang plein et le nombre d'erreurs rang égale à 0. Le coefficient de Gauss défini dans le chapitre 2, équation 2.2 vérifie

$$\begin{bmatrix} \mathrm{N} \\ t \end{bmatrix}_q \leq 4q^{t(\mathrm{N}-t)}.$$

D'après l'équation 2.3, le nombre de matrices de rang égale à t vérifie

$$\mathrm{S}\left(m, \mathrm{N}, q, t\right) \leq 4q^{t(\mathrm{N}-t)}.$$

La probabilité pour qu'une matrice dont les coefficients sont tirés aléatoirement de $\mathbb{F}_q$ soit de rang inférieur à t vérifie

$$\mathrm{P}_e\left(t\right) \leq \frac{q^{t(\mathrm{N}+m-t)}}{q^{m\mathrm{N}}} 4t.$$

Pour t inférieur à $\frac{\mathrm{N}}{2}$ et $m = \mathrm{N}$, la probabilité P_e devient

$$\mathrm{P}_e\left(\frac{\mathrm{N}}{2}\right) \leq q^{-\frac{\mathrm{N}}{2}} 2\mathrm{N}.$$

Pour N assez grand, cette probabilité tend exponentiellement vers zero. Cela implique qu'un décodeur en métrique rang ne peut pas récupérer les paquets source en présence d'erreurs de propagation, d'où la nécessité de trouver une solution pour faire face à ce type d'erreur.

Maintenant, nous calculons la probabilité pour qu'un code en métrique rang arrive à décoder avec succès. Soit r le nombre de colonnes non nulles dans la matrice $\mathbf{E}^{(n)}$.

Soit r le nombre de colonnes non nulles de l'erreur $\mathbf{E}^{(n)}$. La probabilité que le rang de la matrice de taille $(\mathrm{N} \times r)$ des vecteurs colonne non nuls soit inférieur à t vérifie

$$\mathrm{P}_d^{(r)} = \frac{\mathrm{B}\left(\mathrm{N}, r, q, t\right)}{q^{\mathrm{N}r}}, \tag{4.13}$$

où $\mathrm{B}\left(m, r, q, t\right)$ est définie dans l'équation 2.5.

Maintenant, posons p_s le taux d'erreurs symbole du canal. La probabilité pour qu'une erreur se produise dans une colonne de matrice reçu est

$$p_c = 1 - \left(1 - p_s\right)^{\mathrm{N}l}$$

Nous présentons, dans la proposition suivante, la probabilité d'échec du décodage d'un code Gabidulin de paramètres $[\mathrm{N}, k]$.

Proposition 4.2. *Le taux d'erreur paquet d'un code Gabidulin* PER_G *dans un réseau utilisant le codage réseau aléatoire vérifie*

$$\mathrm{PER}_\mathrm{G} = \sum_{i=t+1}^{m} \binom{m}{i} p_c^i \left(1 - p_c\right)^{m-i} \times (1 - \mathrm{P}_d^{(i)})$$

Démonstration. La probabilité que l'erreur se produise dans i colonnes de m est $p_c^i(1-p_c)^{m-i}$. La probabilité que le code en métrique rang ne puisse pas récupérer les erreurs lorsque $i \leq t$ est égal à 0. Lorsque $i > t$, la probabilité que le rang de la matrice soit supérieur à t est $\left(1 - \mathrm{P}_d^{(i)}\right)$, où $\mathrm{P}_d^{(i)}$ est exprimé dans l'équation (4.13). □

Le calcul de la probabilité d'échec du décodage d'un code LRPC se calcule d'une manière différente que celle des codes Gabidulin vu que les codes LRPC sont des codes probabilistes. Donc, il faut prendre en considération les erreurs même dans le cas ou le rang de l'erreur est inférieur à la capacité maximale de décodage.

Proposition 4.3. *Le taux d'erreur paquet d'un code LRPC* PER_L *de paramètres* $[\mathrm{N}, k, d]$ *dans un réseau utilisant le codage réseau aléatoire vérifie*

$$\mathrm{PER}_\mathrm{LRPC} = \sum_{i=0}^{m} \binom{m}{i} p_c^i \left(1 - p_c\right)^{m-i} \left(1 - \sum_{j=0}^{i} \mathrm{P}_\mathrm{R}^{(j)} \frac{\mathrm{S}(\mathrm{N}, i, q, j)}{q^{mi}}\right),$$

où $\mathrm{P}_\mathrm{R}^{(j)}$ *définie dans le chapitre 3, équation 3.2 désigne la probabilité pour que l'algorithme de décodage des codes LRPC récupère avec succès une erreur de rang* j.

Démonstration. On suit le même raisonnement que la démonstration de la proposition précédente. La probabilité que l'erreur se produise dans i colonnes de m est $p_c^i(1-p_c)^{m-i}$. La probabilité que le code LRPC puisse récupérer les i erreurs colonne est la somme des probabilités pour que le code LRPC puisse récupérer une erreur de rang j pour $j \leq i$. Cette probabilité est égale à

$$\mathrm{P}(i) = \sum_{j=0}^{i} \mathrm{P}_{\mathrm{R}}^{(j)} \frac{\mathrm{S}(\mathrm{N}, i, q, j)}{q^{mi}}$$

Donc, la probabilité pour que le code LRPC ne puisse par récupérer une erreur de taille $\mathrm{N} \times i$ est $(1-\mathrm{P}(i))$. □

Pour calculer la probabilité d'échec du décodage en présence de r erreurs rang, il suffit de remplace $\mathrm{P}_d^{(i)}$ par $\mathrm{P}_d^{(i+r)}$ pour le code Gabidulin et $\mathrm{P}_{\mathrm{R}}^{(j)}$ par $\mathrm{P}_{\mathrm{R}}^{(j+r)}$ pour le code LRPC.

Dans le cas de la présence d'un effacement, la matrice de transfert reçue par la destination après avoir appliqué la méthode de résolution des système linéaire est de la forme

$$\tilde{\mathrm{A}} = \begin{bmatrix} 1 & 0 & \cdots & 0 & \tilde{\mathrm{A}}_{1i} & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \tilde{\mathrm{A}}_{2i} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \tilde{\mathrm{A}}_{(i-1)i} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix},$$

où i est le numéro de paquet perdu lors de la transmission. Le rang de la matrice $\tilde{\mathrm{A}} - \mathrm{I}_{\mathrm{N}}$ est 1. Donc, le rang de la différence entre mot de code reçu et le mot de code émis par la source égale à 1. Par conséquent, un effacement peut être considéré comme une erreur rang.

4.3.2 Modèle proposé

Dans cette partie, nous décrivons le modèle proposé pour faire face aux erreurs rang, aux effacements et aux erreurs de propagation. Nous considérons le même scénario, proposé dans la Figure 15, d'une source qui désire transmettre des données à la BS. Pour faire face aux erreurs de type rang et aux erreurs de propagation, nous proposons la concaténation de deux codes, les codes LRPC et les codes convolutifs.

La source utilise le code LRPC pour coder l'ensemble des paquets puis elle applique le code convolutif sur chaque paquet à transmettre, la source dans ce cas se comporte comme N-relais indépendant comme nous pouvons le remarquer dans la Figure 17. Dans ce réseau, on utilise un entrelaceur pseudo-aléatoire pour que les autres nœuds effectuent la même permutation. Sans perte de généralité, nous utilisons un modulateur BPSK. Notre modèle peut facilement être étendu à des niveaux de constellations

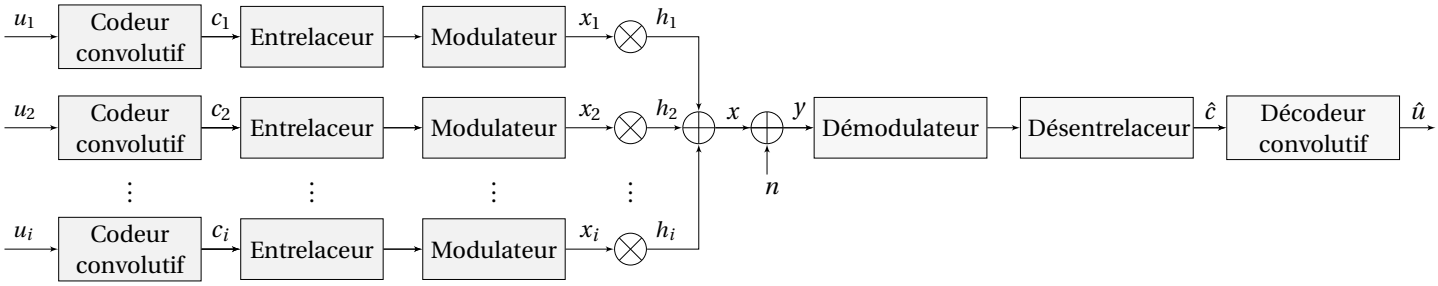


FIGURE 17 – Modèle de transmission dans un réseau de capteurs sans fils utilisant le codage réseau aléatoire et les codes en métrique rang.

d'ordre plus élevé. On suppose que le contrôle de la puissance ainsi que la synchronisation dans tous les nœuds sont parfaits. Chaque nœud reçoit un nombre de paquets et effectue une combinaison de ces paquets. Le nouveau paquet généré est de la forme

$$y = \delta_1 (x_1 h_1 + n_1) + \delta_2 (x_2 h_2 + n_2) + \cdots + \delta_i (x_i h_i + n_i)$$

où δ_j est un paramètre qui indique la présence ou l'absence d'un paquet dans la combinaison effectuée, les h_j sont les coefficients des canaux entre le nœud émetteur et le nœuds récepteur. Le paramètre $n = n_1 + n_2 + \cdots + n_i$ représente le bruit additif gaussien (AWGN) et les paramètres $n_1, n_2, \ldots, n_i$ sont des variables aléatoires Gaussiennes indépendantes de paramètres $\left(0, \frac{N_0}{2}\right)$. Donc, la variable n suit une loi Gaussienne de paramètre $\left(0, \frac{iN_0}{2}\right)$. Chaque relai procède au décodage en utilisant l'algorithme de décodage à maximum de vraisemblance. Si le bruit est très important, le paquet récupéré est forcement erroné ce qui conduit à générer une erreur de type rang.

4.4 Résultats de simulation du modèle proposé

Dans cette section, nous évaluons les performances des codes LRPC par rapport aux codes Gabidulin dans un réseau de capteurs sans fil dans le contexte du codage réseau aléatoire.

4.4.1 Paramètres de simulation

Nous considérons dans nos simulations de nombreux scénarios de transmission. Tout d'abord, à la source, les données sont codées par un code LRPC comme nous l'avons précisé auparavant. Ensuite, un code convolutif avec un rendement de 0.5 est utilisé pour protéger les données transmises entre les relais. Lorsque les données atteignent la destination finale, elles seront décodées à l'aide du décodeur LRPC. Dans le second cas, nous remplaçons la concaténation d'un code LRPC et un code convolutif par la concaténation d'un code Gabidulin et d'un code Reed-Solomon [SRV12]. Nous montrons également les performances des codes Gabidulin utilisés uniquement dans notre modèle de réseau proposé. De plus, nous présentons les résultats de l'approximation théorique pour la probabilité d'erreur du décodage (LRPC, Gabidulin). Pour les codes LRPC et Gabidulin, nous prenons comme paramètres [46,32] dans GF(2^{46}). Par

conséquent, le taux de codage est d'environ 0,66. Pour ces paramètres, les capacités de correction de rang du code LRPC simulé et du code Gabidulin sont 7. Les paramètres utilisés pour le code Reed-Solomon sont $[511, 236]$ sur $\mathbb{F}_{2^9}$. Pour tous les cas, le message initial est une matrice de taille $(m \times k)$ $((46 \times 32)$ dans cette simulation) et le message codé est une matrice de taille $(m \times k)$ $((46 \times 100)$ dans cette simulation). Le rendement du code total est $0,32$.

4.4.2 Résultats de simulation

$\mathrm{LRPC-CC}(i)$ désigne un code LRPC concaténé avec un code convolutif avec i les paquets erronés transmis par les nœuds en panne et th indique un résultat théorique. La même notation est adoptée pour les autres codes. Dans la Figure 18, nous comparons les différents codes en présence du bruit de fond uniquement en raison des erreurs de canal et de l'absence d'erreur de rang. Le canal utilisé est un canal de Rayleigh avec un bruit gaussien AWGN. Nous observons que le $\mathrm{LRPC-CC}$ est meilleur que le $\mathrm{G}abidulin-\mathrm{RS}$ avec un gain de codage de 2 dB. En outre, il surpasse environ 3 dB le code Gabidulin à un PER de 10^{-4}.

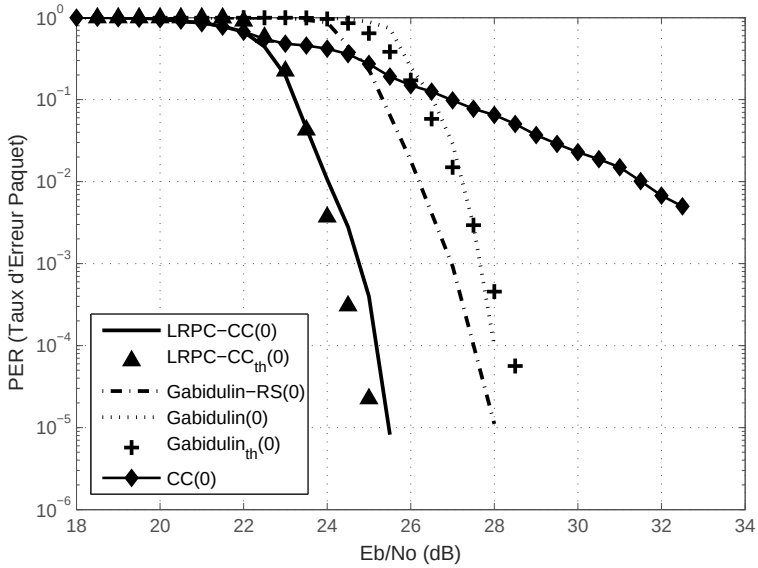


FIGURE 18 – Taux d'erreur de paquets (PER) pour les différents codes avec uniquement du bruit de fond.

La courbe théorique de $\mathrm{LRPC-CC}$ quantifie avec précision la relation entre Eb/No et la probabilité d'erreur de paquet. L'emploi d'un code en métrique rang sans utiliser le code en métrique de Hamming n'a pas de contribution bénéfique à l'égard des erreurs de canal. C'est à cause de la propriété du bruit blanc où chaque symbole a une forte probabilité de générer une erreur de rang et réduit ainsi la capacité de correction d'erreur. En effet, ce résultat est prévisible pour le code Gabidulin puisque les codes en métriques rang sont plus efficaces lorsque les erreurs sont confinées dans les lignes ou dans les colonnes. C'est la raison de l'utilisation d'un code en métrique de Hamming

dans notre cas pour réduire l'impact des erreurs de canal.

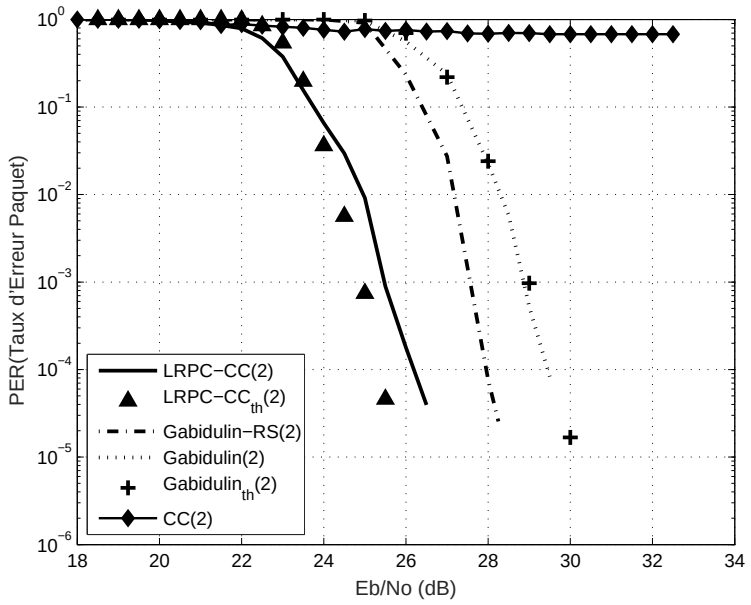


FIGURE 19 – Taux d'erreur de paquets (PER) pour les différents codes avec le bruit de fond et deux paquets erronés injectés sur le réseau.

Dans la Figure 19, nous montrons les performances du réseau avec deux paquets erronés affectés par une défaillance de nœuds. Les performances du code en métrique rang varie légèrement en fonction du nombre de paquets erronés. Les codes concaténés présentent une très bonne performance par rapport au code convolutif CC. En outre, les codes Gabidulin et les codes LRPC décodent parfaitement dans ce dans ce cas. Les performances des codes LRPC concaténés restent meilleurs que celles des codes Gabidulin concaténés. Pour calculer les résultats théoriques, pour le code LRPC – CC et Gabidulin avec 2 paquets erronés, nous avons remplacé $\mathrm{P}_d^{(i)}$ par $\mathrm{P}_d^{(i+2)}$ de la proposition 4.2 pour le code Gabidulin et $\mathrm{P}_\mathrm{R}^{(j)}$ par $\mathrm{P}_\mathrm{R}^{(j+2)}$ de la proposition 4.3 pour le code LRPC.

4.4.3 Comparaison de la complexité

La complexité des codes ne serait pas complète sans une comparaison de leurs complexités de décodage. Le décodage RLNC montre une complexité très importante : $\mathcal{O}(m^3)$, où m est le nombre de paquets générés pour chaque transmission . Pour un codage réseau linéaire, la complexité vaut $\mathcal{O}(m^2)$.

Pour les codes LRPC, si N est le nombre de symboles par mot de code et k le nombre de symboles par message, la complexité globale de l'algorithme est $\mathcal{O}(m(\mathrm{N}-k)^2)$ dans $\mathbb{F}_q$. La complexité d'un décodeur basé sur l'algorithme de Viterbi pour un code convolutif est $\mathcal{O}((\mathrm{N}+m)\sqrt{\mathrm{N}+m})$. La complexité globale du code convolutif, utilisée pour $\mathrm{N}l$ fois, est $\mathcal{O}(\mathrm{N}l(\mathrm{N}+m)\sqrt{\mathrm{N}+m})$. Ainsi, l'algorithme proposé a une

complexité $\mathcal{O}(m(\mathrm{N}-k)^2 + \mathrm{N}l(\mathrm{N}+m)\sqrt{\mathrm{N}+m})$.

Pour les codes de Gabidulin, une modification significative a été faite pour réduire la complexité de la procédure de décodage. La complexité globale de l'algorithme, lorsque $\mathrm{N} = m$, est approximativement $\mathcal{O}(\mathrm{N}m^2(log(m))$ sur $\mathbb{F}_q$. La complexité totale du code Gabidulin-RS est $\mathcal{O}(m^2\mathrm{N}(log(m)+t))$ sur $\mathbb{F}_q$.

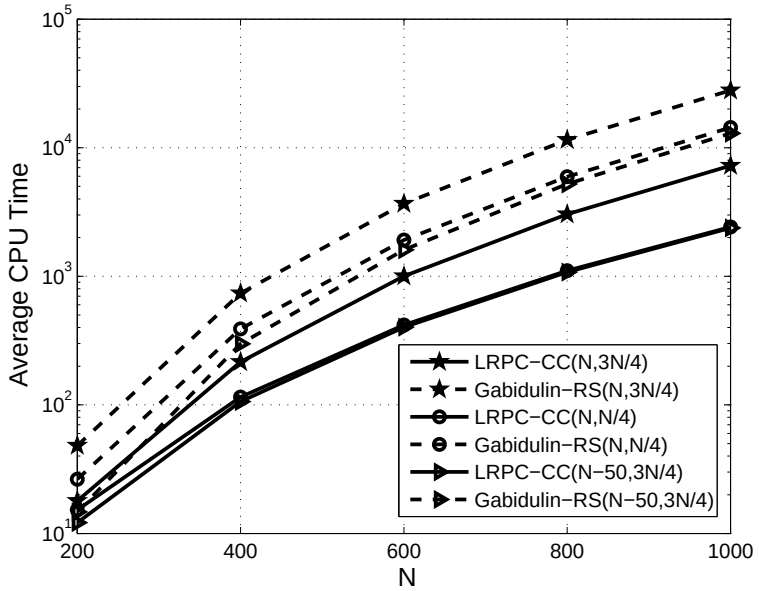


FIGURE 20 – Comparaison de la complexité entre un code LRPC et d'un code Gabidulin pour différentes valeurs de m et k en fonction de N.

Nous présentons les résultats numériques pour les complexités du code LRPC concaténé avec un code convolutif et un code Gabidulin concaténé avec un code Reed-Solomon pour les mêmes paramètres (m, k), en fonction de N, tel que décrit dans la Figure 19. Pour les paramètres de la simulation, nous prenons $l = 200$ et $d = 2$. Nous pouvons observer clairement le gain de performance des codes concaténés proposés en termes de complexité par rapport aux codes Gabidulin-RS.

4.5 Conclusion

Dans ce chapitre, nous avons introduit la technique du codage réseau ainsi que ses différentes variétés pour les réseaux de capteurs sans fil. Ensuite, nous avons présenté quelques scénarios de transmissions utilisant ce type de codage.

Dans la deuxième partie de ce chapitre, nous avons étudié le problème de collecte de données dans les réseaux de capteurs sans fil utilisant le codage réseau en présence de trois types d'erreur, le bruit de fond, les erreurs injectés dans le réseau par un utilisateur malveillant et les effacements qui peuvent être dus aux pannes des nœuds ou aux combinaisons effectuées par les nœuds intermédiaires qui produise un système d'équations incomplet. Nous avons donc proposé un modèle de codage de données

basé sur la concaténation d'un code LRPC avec un code Convolutif. Le code LRPC est utilisé de bout-en-bout, comme code externe, pour la correction des erreurs de type rang. Par conséquent, le décodage est fait par la station de base qui n'a pas de contrainte énergétique. Le code Convolutif est utilisé entre chaque deux relais, comme code interne, pour protéger les données transmises dans le réseau. Nous avons donné l'expression de la probabilité du décodage des codes Gabidulin et des codes LRPC dans le cas du codage réseau aléatoire.

La dernière partie de ce chapitre présente les résultats de simulation du modèle proposé par rapport au modèle basé sur la concaténation d'un code Gabidulin et d'un code Reed-Solomon, utilisé dans la littérature. Nous avons simulé le taux d'erreurs paquet en fonction de SNR dans différents scénarios. Pour les codes LRPC et Gabidulin, nous avons pris comme paramètres $[46, 32]$ dans $GF(2^{46})$. Les résultats de simulation montrent que le modèle proposé est plus performant que le modèle basé sur la concaténation d'un code Gabidulin et d'un code Reed-Solomon. Nous avons également montré que les résultats théoriques sont très proches des résultats de simulation.

Chapitre 5

Codage réseau multi-sources

Aujourd'hui, l'utilisation du codage réseau permet d'améliorer les performances du réseau en termes de débit, de favoriser le passage à l'échelle et d'avoir une sécurité accrue. De plus, il est robuste aux pertes. Nous avons vu, dans le chapitre précédent, l'importance de l'utilisation du codage réseau dans le cas des transmissions unicast et multicast dans un scénario où la source souhaite transmettre des données aux destinations.

Dans un réseau de capteurs sans fil, ce principe reste toujours applicable puisque cela permet d'améliorer les performances du réseau en termes de débit et de latence par rapport aux protocoles traditionnels de store-and-forward. Cependant, l'application du codage réseau dans le cas d'un réseau multi-sources en présence d'erreurs est loin d'être une tâche triviale. Dans de nombreuses applications des réseaux de capteurs sans fil, l'objectif est la transmission de données de plusieurs sources vers la station de base (BS) à travers des relais ayant une topologie arbitraire. Ce scénario est appelé collecte de données dans les réseaux de capteurs, (voir la Figure 1).

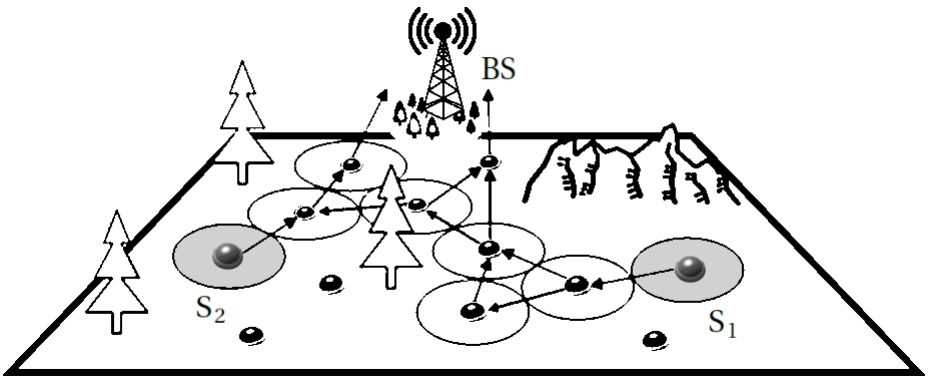


FIGURE 1 – Scénario de collecte de données.

Dans ce chapitre, nous nous intéressons au problème de collecte de données par plusieurs sources vers une station de base. Nous considérons deux cas de codage réseau dans un scénario de collecte de données. Le premier cas est le codage réseau cohérent où la topologie du réseau ainsi que la matrice de transfert sont supposées

être connues par la station de base. Dans ce cas les combinaisons ainsi que les routes pour acheminer les données sources sont choisies de manière optimale. Le second cas est le codage réseau non cohérent. Dans ce cas la station de base n'a aucune connaissance sur la topologie du réseau. Cela peut être dû à des contrainte de mobilité, comme dans le cas des réseau de véhicules, ou à des changements d'état des nœuds (marche / veille) d'un réseau pour préserver l'énergie le plus longtemps possible comme dans le cas de collecte de données.

5.1 Description du code

Dans le chapitre 2 nous avons introduit les codes en métrique rang relevés. La définition suivante introduit la notion *d'un code s-relevé*.

Définition 5.1. Code relevé
Soit $\mathcal{C}$ un code en métrique rang de longueur N, de distance minimale d et de dimension k sur $\mathbb{F}_{q^m}$. Soit $\boldsymbol{c}$ un mot de code de $\mathcal{C}$ et $\mathbf{C}$ sa représentation matricielle dans une base fixe β. Un code $\mathcal{C}^{(1)}$ est dit relevé de code $\mathcal{C}$ s'il vérifie la relation

$$\mathcal{C}^{(1)} = \left\{ \mathbf{C}^{(1)} \quad : \quad \mathbf{C}^{(1)} = \begin{pmatrix} \mathrm{I_N} & \mathbf{C}_1^{\mathsf{T}} \end{pmatrix} \quad et \quad \mathbf{C} \in \mathbb{F}_q^{m\times \mathrm{N}} \right\}. \tag{5.1}$$

Définition 5.2. Code s-relevé
Soit $\{\mathcal{C}_{i=1:s}\}$ une famille de code en métrique rang de longueur N, de distance minimale d et dimension k sur $\mathbb{F}_{q^m}$. Soit $\boldsymbol{c}_i$ un mot de code de $\mathcal{C}_i$ pour $i = 1 : s$. Un code $\mathcal{C}^{(s)}$ est dit s-relevé des codes $\mathcal{C}_{i=1:s}$ s'il vérifie

$$\mathcal{C}^{(s)} = \left\{ \mathbf{C}^{(s)} \quad : \quad \mathbf{C}^{(s)} = \begin{bmatrix} \mathrm{I_N} & 0_\mathrm{N} & \cdots & 0_\mathrm{N} & \mathbf{C}_1^{\mathsf{T}} \\ 0_\mathrm{N} & \mathrm{I_N} & \cdots & 0_\mathrm{N} & \mathbf{C}_2^{\mathsf{T}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_\mathrm{N} & 0_\mathrm{N} & \cdots & \mathrm{I_N} & \mathbf{C}_s^{\mathsf{T}} \end{bmatrix} \quad et \quad \mathbf{C}_1, \cdots, \mathbf{C}_s \in \mathbb{F}_q^{m\times \mathrm{N}} \right\}. \tag{5.2}$$

Les matrices $\mathbf{C}_{i=1:s}$ sont les représentations matricielles des mots de codes $\boldsymbol{c}_{i=1:s}$ dans une base fixée β. Par conséquent, les matrices $\mathbf{C}$ sont des éléments du corps $\mathbb{F}_q^{s\mathrm{N}\times(s\mathrm{N}+m)}$. Les matrices $\mathrm{I_N}$ et 0_N sont respectivement la matrice identité, de taille $\mathrm{N}\times\mathrm{N}$, et la matrice nulle de taille $\mathrm{N}\times\mathrm{N}$. Pour simplifier les notations, nous avons pris la même taille pour tous les codes $\mathcal{C}_{i=1:s}$. Notons que le code sous espace construit à partir de s matrices relevées est de dimension constante.

Nous avons donné une forme simple aux éléments du code $\mathcal{C}^{(s)}$ pour mieux comprendre la construction de ce dernier. Dans la pratique, nous appliquons à chaque mot

de code de $\mathcal{C}$ la fonction *s-lifting* pour ajouter la matrice identité et des zéros. La fonction *s-lifting* Λ_i est définie comme suit

$$\begin{aligned} \Lambda_i^{(s)}: \quad & \mathcal{C} \to \mathbb{F}_q^{\mathrm{N}\times(s\mathrm{N}+m)} \\ & \boldsymbol{c} \mapsto \Lambda_i^{(s)}(\boldsymbol{c}) = \big(\underbrace{0_\mathrm{N} \quad \cdots \quad 0_\mathrm{N}}_{i-1} \quad \mathrm{I}_\mathrm{N} \quad \underbrace{0_\mathrm{N} \quad \cdots \quad 0_\mathrm{N}}_{s-i} \quad \mathbf{C}^\mathsf{T}\big), \end{aligned} \tag{5.3}$$

Nous nous intéressons maintenant à la définition de la distance d'injection introduite par Silva et al. dans [SKK08].

Définition 5.3. Distance d'injection
La distance d'injection entre deux matrices M_1 et M_2 est définie par :

$$d_\mathrm{I}(\mathrm{M}_1, \mathrm{M}_2) = rang\begin{pmatrix}\mathrm{M}_1\\ \mathrm{M}_2\end{pmatrix} - \min\big(rang(\mathrm{M}_1), rang(\mathrm{M}_2)\big).$$

S'il n'y a pas d'ambiguïté, nous utilisons définition précédente pour des vecteurs ou des mots de codes. Les matrices M_1 et M_2 sont dans ce cas les représentations de ces vecteurs dans une base fixe. En effet, on prend le transposé des deux vecteurs, ensuite on les transforme en matrices.

La proposition suivante montre le lien entre la distance d'injection et la distance rang [Sil09, Proposition 4.23].

Proposition 5.1.
Soit $\mathcal{C}$ un code en métrique rang et soient $\boldsymbol{c}_1$ et $\boldsymbol{c}_2$ deux mot du code $\mathcal{C}$. Alors, on a

$$d_\mathrm{I}(\Lambda(c_1), \Lambda(c_2)) = d_\mathrm{R}(c_1, c_2),$$

où d_I est la distance d'injection, D_R est la distance rang et $\Lambda = \Lambda_1^{(1)}$.

Cette relation, qui lie un code en métrique rang à un code relevé, est importante pour les démonstrations des propositions suivantes.

Proposition 5.2.
Soit $\mathcal{C}$ un code en métrique rang et soient $\boldsymbol{c}_1$ et $\boldsymbol{c}_2$ deux mots du code $\mathcal{C}$. Alors, on a

$$d_\mathrm{I}\left(\Lambda_i^{(s)}(c_1), \Lambda_j^{(s)}(c_2)\right) = \begin{cases} d_\mathrm{R}(c_1, c_2), & si\ i = j \\ \mathrm{N}, & sinon \end{cases}$$

Démonstration. La première assertion est déduite à partir de la proposition précédente.
Pour simplifier, on prend $s = 2$, $i = 1$ et $j = 2$.

$$\begin{aligned} d_\mathrm{I}\left(\Lambda_i^{(s)}(c_1), \Lambda_j^{(s)}(c_2)\right) &= rang\begin{pmatrix}\mathrm{I}_\mathrm{N} & 0_\mathrm{N} & \mathbf{C}_1\\ 0_\mathrm{N} & \mathrm{I}_\mathrm{N} & \mathbf{C}_2\end{pmatrix} - \mathrm{N} \\ &= \mathrm{N} \end{aligned}$$

□

On déduit de la proposition précédente que dans le cas où i est différent de j, l'intersection des espaces générés par les matrices $\Lambda_i^{(s)}(c_1)$ et $\Lambda_j^{(s)}(c_2)$ est égale au vecteur nul. Dans ce cas la superposition des deux codes est faisable.

5.1.1 Codage réseau cohérent

Supposons qu'une source transmette une matrice codée C vers une destination en utilisant le codage réseau aléatoire. La matrice de transfert A dans ce cas est supposée être connue par la destination. Cette matrice est de taille $\mathrm{N}' \times \mathrm{N}$. De plus, on suppose qu'un attaquant a la capacité d'introduire r paquets dans le réseau. La destination reçoit Y qui est de la forme

$$\mathrm{Y} = \mathrm{AC}^{\intercal} + \mathrm{E}. \tag{5.4}$$

La matrice E est la matrice erreur dans $\mathbb{F}_q^{\mathrm{N}' \times \mathrm{N}}$ qui vérifie $\mathrm{E} = \mathrm{DZ}$ où D est une matrice de taille $\mathrm{N}' \times r$ et Z est une matrice de taille $r \times m$. La matrice D représente l'ensemble des combinaisons effectuées sur les r paquets injectés dans le réseau lors de la transmission, autrement dit sur Z.

Pour simplifier les notations utilisées dans cette partie, on suppose que la matrice E est de rang r. Si le rang de E est égale à $r' \leq r$, l'espace engendré par les lignes de la matrice D' est égale à l'espace engendré par les lignes de la matrice D, où la matrice D' de taille $\mathrm{N} \times r'$ vérifie $\mathrm{E} = \mathrm{D}'\mathrm{Z}'$, et Z' est une matrice de taille $r' \times m$. Dans ce cas le nombre d'erreurs injectés par l'attaquant est r' et non pas r. Pour résumer, nous considèrerons le pire scénario où l'attaquant n'injecte que des vecteurs linéairement indépendants.

Le mot de code le plus proche du mot C envoyé par la source, sachant que la destination reçoit Y, vérifie

$$\widehat{\mathrm{C}} = arg \min_{c \in \mathcal{C}} \Big(\min_{\substack{\mathrm{D} \in \mathbb{F}_q^{\mathrm{N} \times r}, \mathrm{Z} \in \mathbb{F}_q^{r \times m}: \\ \mathrm{Y} = \mathrm{AC}^{\intercal} + \mathrm{DZ}}} r \Big). \tag{5.5}$$

On rappelle que le rang d'une matrice E vérifie

$$rang\,(\mathrm{E}) = \min_{\substack{\mathrm{D} \in \mathbb{F}_q^{\mathrm{N}' \times r}, \mathrm{Z} \in \mathbb{F}_q^{r \times m}: \\ \mathrm{E} = \mathrm{DZ}}} r. \tag{5.6}$$

Par conséquent, le mot de code le plus proche de C vérifie

$$\widehat{\mathrm{C}} = arg \min_{c \in \mathcal{C}} \left(rang \left(\mathrm{Y} - \mathrm{AC}^{\intercal} \right) \right). \tag{5.7}$$

Si de plus la matrice A est inversible, l'équation précédente vérifie

$$\widehat{c} = arg \min_{c \in \mathcal{C}} \left(d_{\mathrm{R}} \left(\widehat{y}, c \right) \right),$$

où $\widehat{y}$ désigne la réécriture de la matrice $\mathrm{A}^{-1}\mathrm{Y}$ dans le corps $\mathbb{F}_{q^m}$. Ce résultat est important et montre l'importance de l'utilisation des codes en métrique rang dans le cas du codage réseau cohérent.

Soit $\mu_c = \mathrm{N} - rang\,(\mathrm{A})$ le nombre d'effacements colonne. La proposition suivante donne une idée sur la capacité de correction d'un code en métrique rang dans le cas du codage réseau [Sil09, Proposition 4.8].

Proposition 5.3. *Soit $\mathcal{C}$ un code en métrique rang de paramètre* $[\mathrm{N}, k, d]$. *Le code $\mathcal{C}$ peut décoder jusqu'à* $\left\lfloor \frac{d-1-\mu_c}{2} \right\rfloor$ *erreurs rang.*

La condition nécessaire pour que le nombre d'erreurs soit inférieur à $\left\lfloor \frac{d-1-\mu_c}{2} \right\rfloor$ devient nécessaire et suffisante si $\mathcal{C}$ est un code MDR et $m \geq \mathrm{N}$.

5.1.2 Codage réseau non cohérent

Considérons un scénario de transmission dans un réseau qui utilise le codage réseau non cohérent. La différence entre le scénario présenté dans cette partie et celui de partie précédente est que la matrice de transfert A n'est pas connue par la destination. Dans ce cas, les combinaisons effectués par les nœuds intermédiaires doivent être envoyées vers la destination comme nous l'avons expliqué dans le chapitre précédent. La deuxième différence avec le scénario considéré dans la partie précédente est que l'attaquant choisit également les entêtes des paquets transférés. Autrement dit, l'attaquant peut modifier également la matrice de transfert.

Soit $\mathcal{C}$ un code en métrique rang de paramètres $[\mathrm{N}, k]$. Supposons que la matrice transmise par la source est $\mathrm{C}^{(1)}$ relevé du mot de code C. Donc, à la place de transmettre C, la source transmet $\Lambda_1^1(\mathrm{C})$. Chaque paquet transmis est de longueur $m+\mathrm{N}$. La destination reçoit Y, qui est cette fois-ci une concaténation de $\widehat{\mathrm{A}}$ et $\mathrm{AC}^{(1)} + \mathrm{DZ}$, où $\widehat{\mathrm{A}}$ de taille $\mathrm{N}' \times \mathrm{N}$ est la matrice de transfert erronée. Le problème (5.5) devient

$$\widehat{\mathrm{C}} = arg \min_{c \in \mathcal{C}} \Big(\min_{\substack{\mathrm{A} \in \mathbb{F}_q^{\mathrm{N}' \times \mathrm{N}}, \\ \mathrm{D} \in \mathbb{F}_q^{\mathrm{N} \times r}, \mathrm{Z} \in \mathbb{F}_q^{r \times m}: \\ \mathrm{Y} = \mathrm{AC}^{\intercal} + \mathrm{DZ}}} r \Big), \tag{5.8}$$

où A n'est pas forcement la matrice de transfert, mais plutôt une matrice qui est la plus proche en terme de rang de la matrice de transfert reçu par la destination.

Dans le processus de décodage on utilise la métrique d'injection, sachant que la source génère un mot de code en métrique rang relevé, et on a le résultat suivant

Proposition 5.4. *[Sil09, Théorème 4.20]*
Soit $\mathcal{C}$ un code en métrique rang de paramètre $[\mathrm{N}, k, d]$. *Le code $\mathcal{C}$ peut décoder jusqu'à* $\left\lfloor \frac{d_{\mathrm{I}}-1-\mu_c}{2} \right\rfloor$ *erreurs rang. La distance d_{I} représente la distance minimale d'injection du code* $\mathrm{C}^{(1)}$.

Cette proposition montre l'importance de l'utilisation de la métrique d'injection dans le cas du codage réseau non cohérent.

Pour montrer la relation entre le problème 5.5 et le problème 5.8, on suppose que

la matrice Y reçu est $\begin{bmatrix}\widehat{\mathrm{A}} & \mathrm{X}\end{bmatrix}$. On a

$$\begin{aligned} d_{\mathrm{I}}(\Lambda(\mathrm{C}),\mathrm{Y}) &= rang\begin{bmatrix}\mathrm{I_N} & \mathrm{C}\\ \widehat{\mathrm{A}} & \mathrm{X}\end{bmatrix} - \min\{\mathrm{N},\mathrm{N}'\}\\ &= rang\begin{bmatrix}\mathrm{I_N} & \mathrm{C}^{\intercal}\\ 0 & \mathrm{X}-\widehat{\mathrm{A}}\mathrm{C}^{\intercal}\end{bmatrix} - \min\{\mathrm{N},\mathrm{N}'\}\\ &= rang\left(\mathrm{X}-\widehat{\mathrm{A}}\mathrm{C}^{\intercal}\right) - \min\{0,\mathrm{N}'-\mathrm{N}\}\end{aligned}$$

Puisque le terme $\min\{0,\mathrm{N}'-\mathrm{N}\}$ est constant, l'équation précédente est équivalente à

$$\widehat{\mathrm{C}} = arg\min_{\mathrm{C}\in\mathcal{C}}\left(rang\left(\mathrm{Y}-\widehat{\mathrm{A}}\mathrm{C}^{\intercal}\right)\right). \tag{5.9}$$

Si on compare les deux equations (5.7) et (5.5) on déduit que le problème de codage réseau non cohérent est équivalent au problème de codage réseau cohérent. Le fait que la matrice A est supposée être connue par la destination dans le cas du codage réseau cohérent est équivalent à dire que l'attaquant injecte des valeurs nulles dans l'entête qui corresponds à A.

5.1.3 Décodage des codes en métrique rang

Le problème de décodage des codes en métrique rang dans le cas d'un réseau aléatoire a été résolu dans plusieurs travaux [YC+06] [CY+06] [Mat07]. La manière la plus simple de le faire dans le cas du codage réseau cohérent, est de considérer un code $\mathcal{C}_{\mathrm{A}}$ construit à partir de l'image par A des éléments du code en métrique rang $\mathcal{C}$, c'est à dire $\mathcal{C} = \{\mathrm{AC}^{\intercal}, \mathrm{C}\in\mathcal{C}\}$. Nous allons voir, dans la suite, que cette méthode est peu efficace en présence des effacements ligne et colonne car cette méthode considère que les effacements sont aussi des erreurs rang. C'est la même méthode que nous avons appliqué dans le chapitre 4 pour la récupération des paquets envoyés par la source en présence des effacement.

Nous présentons dans cette partie un algorithme de décodage efficace proposé par Silva, Kschischang et Ralf dans [SKK08]. Soient $\mu_c = \mathrm{N} - rang\,\widehat{\mathrm{A}}$ et $\mu_l = \mathrm{N}' - rang\,\widehat{\mathrm{A}}$ respectivement les effacements colonne et ligne de la matrice $\widehat{\mathrm{A}}$. Rappelons que Y s'écrit sous la forme de la concaténation de la matrice $\widehat{\mathrm{A}}$ et X.

$$\mathrm{Y} = \left[\begin{array}{cccc|cccc} \mathrm{A}_{11} & \mathrm{A}_{12} & \cdots & \mathrm{A}_{1\mathrm{N}} & \mathrm{X}_{11} & \mathrm{X}_{12} & \cdots & \mathrm{X}_{1m}\\ \mathrm{A}_{21} & \mathrm{A}_{22} & \cdots & \mathrm{A}_{2\mathrm{N}} & \mathrm{X}_{21} & \mathrm{X}_{22} & \cdots & \mathrm{X}_{2m}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathrm{A}_{\mathrm{N}'1} & \mathrm{A}_{\mathrm{N}'2} & \cdots & \mathrm{A}_{\mathrm{N}''\mathrm{N}} & \mathrm{X}_{\mathrm{N}'1} & \mathrm{X}_{\mathrm{N}'2} & \cdots & \mathrm{X}_{\mathrm{N}'m}\end{array}\right]$$

On définit une fonction qui transforme une matrice B en une matrice $\tilde{\mathrm{B}}$ qui soit la plus proche de la matrice identité en terme du poids de Hamming. Elle sera notée $\mathrm{TI}d_n$ où n est le nombre de ligne de la matrice $\tilde{\mathrm{B}}$:

$$\tilde{\mathrm{B}} = \mathrm{TI}d_n(\mathrm{B}) = arg\,min_{\mathrm{D}}\left(w\left(\mathrm{DB} = \tilde{\mathrm{I}}_n\right)\right)$$

Si la matrice $\tilde{\mathrm{I}}_n$ n'est pas carrée, on complète les lignes ou les colonnes restantes avec des zéros. Ainsi, La matrice $\tilde{\mathrm{Y}}$ s'écrit sous la forme

$$\tilde{\mathrm{Y}} = \begin{bmatrix} \mathrm{J} & \tilde{\mathrm{X}} \\ 0 & \mathrm{V} \end{bmatrix} \tag{5.10}$$

$$\tilde{\mathrm{Y}} = \left[\begin{array}{cccccccc|cccc}
1 & 0 & \cdots & 0 & \tilde{\mathrm{Y}}_{1i} & 0 & \cdots & 0 & \tilde{\mathrm{X}}_{11} & \tilde{\mathrm{X}}_{12} & \cdots & \tilde{\mathrm{X}}_{1m} \\
0 & 1 & \cdots & 0 & \tilde{\mathrm{Y}}_{2i} & 0 & \cdots & 0 & \tilde{\mathrm{X}}_{21} & \tilde{\mathrm{X}}_{22} & \cdots & \tilde{\mathrm{X}}_{2m} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & \tilde{\mathrm{X}}_{(i-1)1} & \tilde{\mathrm{X}}_{(i-1)2} & \cdots & \tilde{\mathrm{X}}_{(i-1)m} \\
0 & 0 & \cdots & 0 & 1-1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & \tilde{\mathrm{X}}_{(i+1)1} & \tilde{\mathrm{X}}_{(i+1)2} & \cdots & \tilde{\mathrm{X}}_{(i+1)m} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & \tilde{\mathrm{X}}_{\mathrm{N}1} & \tilde{\mathrm{X}}_{\mathrm{N}2} & \cdots & \tilde{\mathrm{X}}_{\mathrm{N}m} \\
\hline
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \tilde{\mathrm{X}}_{(\mathrm{N}+1)1} & \tilde{\mathrm{X}}_{(\mathrm{N}+1)2} & \cdots & \tilde{\mathrm{X}}_{(\mathrm{N}+1)m} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \tilde{\mathrm{X}}_{(\mathrm{N}+\mu_l)1} & \tilde{\mathrm{X}}_{(\mathrm{N}+\mu_l)2} & \cdots & \tilde{\mathrm{X}}_{(\mathrm{N}+\mu_l)m}
\end{array}\right]$$

La matrice V de taille $\mu_l \times m$ représente les effacements lignes du code. Nous montrons dans la suite que cette matrice est très importante pour récupérer une partie de la base de l'erreur dans le cas des codes LRPC.

Le paramètre i dans cette matrice représente la colonne effacée de la matrice A. Pour simplifier, nous considérons le cas $\mu_c = 1$. On note u l'ensemble qui vérifie $u = \{i : \mathrm{J}_{ii} = 0\}$. Le cardinale de cette ensemble est μ_c. On associe à u l'ensemble $u_c = \{1, 2, \cdots, \mu_c\}$. Nous choisissons intentionnellement les éléments de u de la matrice J pour qu'ils soient $1-1$ et non pas zéros pour dire que la matrice J s'écrit sous la forme $\mathrm{J} = \mathrm{I}_\mathrm{N} + \mathrm{L}\mathrm{D}_u$,où L est une matrice de taille $\mathrm{N} \times \mu_c$ et D_u est une matrice de taille $\mu_c \times \mathrm{N}$ qui vérifie

$$\mathrm{D}_{u(ij)} = \begin{cases} 1, & \text{si } (i,j) \in u_c \times u \\ 0, & \text{sinon} \end{cases}$$

Au final, $\tilde{\mathrm{Y}}$ s'écrit sous la forme

$$\tilde{\mathrm{Y}} = \begin{bmatrix} \mathrm{I}_\mathrm{N} + \mathrm{L}\mathrm{D}_u & \tilde{\mathrm{X}} \\ 0 & \mathrm{V} \end{bmatrix} \tag{5.11}$$

On appelle $(\tilde{\mathrm{X}}, \mathrm{V}, \mathrm{L})$ la réduction de la matrice Y. Le résultat le plus important de cette section est introduit dans le théorème suivant [Sil09, Théorème 5.1].

Théorème 5.1. *Soit* $(\tilde{\mathrm{X}}, \mathrm{V}, \mathrm{L}) \in \mathbb{F}_q^{\mathrm{N}\times m} \times \mathbb{F}_q^{\mu_l \times m} \times \mathbb{F}_q^{\mathrm{N}\times\mu_c}$ *la réduction de la matrice* Y, *on a*

$$d_\mathrm{I}(\Lambda(\mathrm{C}), \mathrm{Y}) = rang \begin{bmatrix} \mathrm{L} & \tilde{\mathrm{X}} - \mathrm{C} \\ 0 & \mathrm{V} \end{bmatrix} - \min\{\mu_l, \mu_c\}$$

Le mot de code le plus proche au mot reçu Y vérifie

$$\widehat{\mathrm{C}} = arg \min_{\mathrm{C}\in\mathcal{C}} \left(rang \begin{bmatrix} \mathrm{L} & \tilde{\mathrm{X}} - \mathrm{C} \\ 0 & \mathrm{V} \end{bmatrix} \right). \tag{5.12}$$

D'une manière analogue aux codes en métrique de Hamming, le théorème suivant montre la capacité de correction d'un code en métrique rang face à r erreurs rang, μ_l effacement ligne et μ_c effacement colonne [Sil09, Théorème 5.1].

Théorème 5.2. *Soit $\mathcal{C}$ un code en métrique rang de paramètres* $[\mathrm{N}, k, d]$. *Le code $\mathcal{C}$ est capable de corriger r erreurs rang, μ_l effacement ligne et μ_c effacement colonne si et seulement si* $2r + \mu_l + \mu_c \leq d - 1$.

Ce théorème montre qu'en prenant en compte les effacements lignes et colonnes, la capacité de correction augmente considérablement. Si on ne prend pas en considération cette remarque, les effacements serons considérés comme des erreurs rang et la capacité de correction sera $\frac{d-1}{2} - \mu_l - \mu_c$.

5.1.4 Décodage des codes LRPC

Nous considérons dans cette partie les codes LRPC, en particulier les codes LRPC matriciels définis dans les chapitre 2. Nous choisissons ce type de codes pour ses performances en terme de complexité et de capacité de correction. De plus, ces codes ont des propriétés particulières qui les caractérisent par rapport aux codes MRD.

On se place dorénavant dans le corps fini $\mathbb{F}_q$. Soit $\mathcal{C}$ un code en métrique rang de paramètres $\left[m\mathrm{N}, mk, \hat{d}\right]$. On reprend les notations du paragraphe précédent, le mot de code le plus proche du mot reçu Y est $\widehat{\mathrm{C}}$ qui vérifie l'équation 5.5, et $\left(\tilde{\mathrm{X}}, \mathrm{V}, \mathrm{L}\right)$ est la réduction de la matrice Y. On pose $\mathrm{E} = \tilde{\mathrm{X}} - \widehat{\mathrm{C}}$ l'erreur rang après avoir appliqué la réduction de Y. Il est évident que E dans ce cas n'est pas forcement de rang faible. Autrement dit, il peut exister une autre matrice d'erreur $\mathrm{E}^{(1)}$ telle que $rang\left(\mathrm{E}^{(1)}\right) < rang\,(\mathrm{E})$ et $\mathrm{E}^{(1)} = \tilde{\mathrm{X}} - \widehat{\mathrm{C}}^{(1)}$ avec $\widehat{\mathrm{C}}^{(1)}$ est un mot de code $\mathcal{C}$.

En faisant appel à l'équation 5.5, on pose $\widehat{\mathrm{E}} = \tilde{\mathrm{X}} - \widehat{\mathrm{C}}$ et

$$\hat{r} = rang \begin{bmatrix} \mathrm{L} & \widehat{\mathrm{E}} \\ 0 & \mathrm{V} \end{bmatrix}.$$

On note $\widehat{\mathrm{V}}$ une matrice générée par les lignes de la matrice V vérifiant $\langle\widehat{\mathrm{V}}\rangle = \langle\mathrm{V}\rangle \cap \langle\widehat{\mathrm{E}}\rangle$, et V_c une matrice générée par les lignes de la matrice V vérifiant $\{0\} = \langle\mathrm{V}_c\rangle \cap \langle\widehat{\mathrm{E}}\rangle$ où l'espace $\langle\mathrm{V}_c\rangle$ est de dimension μ_c^c. On a

$$\widehat{\mathrm{C}} = arg \min_{\mathrm{C}\in\mathcal{C}} \left(rang \begin{bmatrix} \mathrm{L} & \tilde{\mathrm{X}} - \mathrm{C} \\ 0 & \widehat{\mathrm{V}} \end{bmatrix} \right). \tag{5.13}$$

En effet, la matrice erreur $\widehat{\mathrm{E}}$ vérifie

$$\hat{r} = rang \begin{bmatrix} L & \widehat{E} \\ 0 & \widehat{V} \\ 0 & V_c \end{bmatrix} \Leftrightarrow \hat{r} - \mu_c^c = rang \begin{bmatrix} L & \widehat{E} \\ 0 & \widehat{V} \end{bmatrix}$$
$$\Leftrightarrow \hat{r} - \mu_c^c = rang \begin{bmatrix} L & \widehat{E} \end{bmatrix}$$

Supposons que un mot de code $\widehat{C}_1$ vérifie

$$\widehat{C}_1 = arg \min_{C \in \mathcal{C}} \left(rang \begin{bmatrix} L & \tilde{X} - C \\ 0 & \widehat{V} \end{bmatrix} \right). \tag{5.14}$$

et

$$\hat{r}_1 = rang \begin{bmatrix} L & \tilde{X} - \widehat{C}_1 \\ 0 & \widehat{V} \end{bmatrix}.$$

On a $\hat{r}_1 < \hat{r} - \mu_c^c$, donc

$$\hat{r} > \hat{r}_1 + \mu_c^c \geq rang \begin{bmatrix} L & \tilde{X} - \widehat{C}_1 \\ 0 & \widehat{V} \\ 0 & V_c \end{bmatrix},$$

ce qui est absurde puisque $\hat{r}$ est le plus petit rang qui vérifie l'équation 5.5. On en déduit que l'équation (5.5) et l'équation (5.8) sont équivalentes, cela veut dire que même si un élément de $\langle V \rangle$ n'appartient pas au support de l'erreur, ceci n'a aucun effet sur le choix de C.

Comme nous avons montré dans la section section 2 du chapitre 2, l'étape la plus importante de l'algorithme de décodage des codes LRPC matricielles est la récupération de support de l'erreur. Cette étape est également la plus couteuse de l'algorithme de décodage des codes LRPC en terme de complexité et c'est l'étape qui a le poids le plus important dans la formule de la probabilité d'échec. Ainsi, il est nécessaire de réussir la récupération du support d'erreur pour réussir le décodage.

Supposons que la matrice L est nulle et la dimension de l'espace produit $F.\left(\widehat{E} + V\right)$ égale $d.dim\left(\widehat{E} + V\right)$, l'équation 5.5 devient

$$\widehat{C}_1 = arg \min_{C \in \mathcal{C} E_0 = \langle \widehat{V} \rangle} \left(rang \left[\tilde{X} - C \right] \right),$$

où E_0 est l'initialisation du support de l'erreur dans l'algorithme de décodage des codes LRPC.

L'étape de l'initialisation du syndrome dans l'algorithme de décodage des codes LRPC est précédé par une initialisation du syndrome par l'espace produit $\langle FV \rangle$, ce qui garantit que l'espace V soit dans le support de l'erreur. Comme nous avons expliqué dans la partie des codes LRPC étendu du chapitre 2, le fait d'initialiser l'erreur par V donne plus de chance de récupérer les autres éléments de la base du support de l'erreur lors du calcul des intersections.

5.2 Description du modèle

Considérons un réseau constitué de s sources qui désirent envoyer des données à la station de base. Chaque source dispose d'un message à transmettre à la station de base. Ce dernier est découpé pour générer k paquets de taille m (on peut ajouter des zéros si besoin). La source S_i encode les paquets générés en utilisant un code en métrique rang de paramètre $[\mathrm{N}, k]$, et elle applique le s-lifting et transmet les paquets codés aux relais. Notons que le codage réseau peut être aussi effectué au sein des nœuds sources en ajoutant la redondance. Cela évite les pertes lors de la transmission des paquets entre les sources et leurs relais voisins. De plus, il peut éviter les effacements lors des transmissions des paquets entre les relais ou entre les relais et la destination.

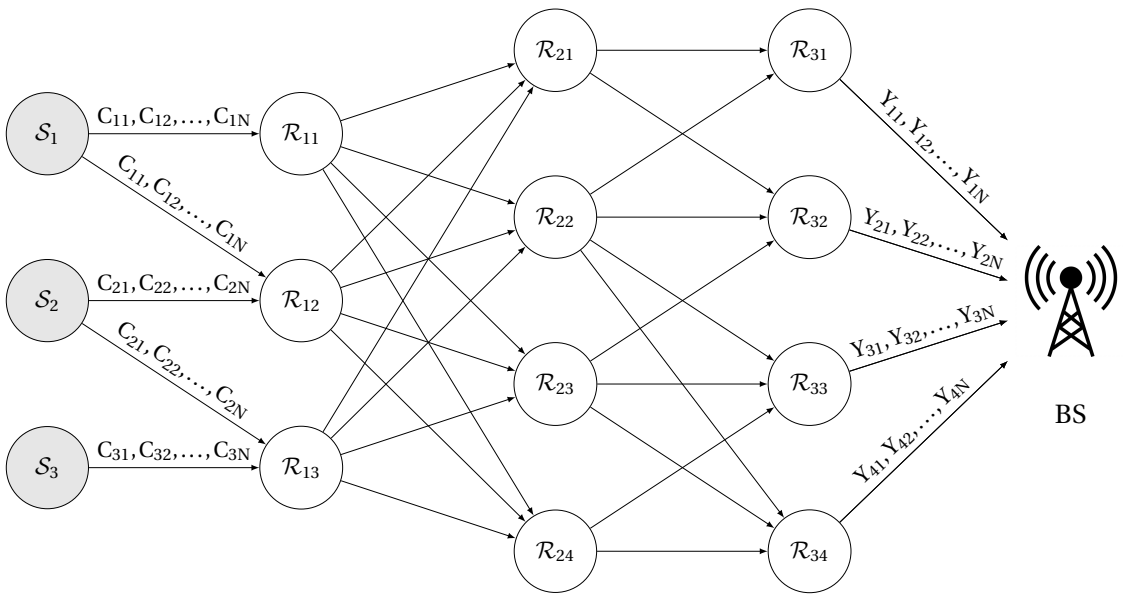


FIGURE 2 – Transmission de données depuis 3 sources vers la BS.

Chaque relais qui reçoit les paquets source effectue le codage réseau aléatoire et les envoie aux autres relais jusqu'à ce que la station de base reçoive les paquets sources comme nous pouvons le voir dans la figure 2. Notons que le nombre de paquets reçus par la destination peut être différent (plus grand ou plus petit) du nombre de paquets transmises par les sources. Cela dépend de la topologie du réseau ainsi que de la manière dont les combinaisons sont effectuées. La station de base reçoit $\sum_{i=1}^{s} \mathrm{N}_i$ paquets sources. Nous simplifions le travail en considérant seulement le cas où les N_i sont tous égaux. L'étude de ce cas spécial donne le même résultat que le cas général. Supposons dans ce cas qu'on a reçu $y_{11}, y_{12}, ..., y_{s\mathrm{N}'}$ que nous pouvons exprimer en s blocs de matrices de taille $\mathrm{N}' \times m\,(s\mathrm{N} + m)$ comme suit

$$\mathrm{Y} = \begin{bmatrix} \mathrm{Y}_1 \\ \mathrm{Y}_2 \\ \vdots \\ \mathrm{Y}_s \end{bmatrix}.$$

Afin de limiter l'impact du bruit de fond, nous utilisons un outil de détection d'erreur, le contrôle de redondance cyclique (CRC) par exemple, dans les nœuds intermédiaires. Chaque nœud vérifie l'intégrité des paquets reçus et rejette les paquets corrompus. Supposons que l'attaquant connait les paramètres du contrôle de redondance cyclique. Ce dernier génère des faux paquets et les injecte dans les réseaux.

Supposons que r paquets erronés soient introduits dans le réseau durant la transmission des paquets sources. Autrement dit, le rang total de la matrice erreur finale est égal à r. Puisque les combinaisons sont faites de manière aléatoire et que les sources transmettent d'une manière aléatoire et uniforme leurs paquets, les erreurs affectent presque tous les paquets sources avec une grande probabilité. Pour cette raison, nous avons utilisé un code en métrique rang. Au niveau de la station de base, les paquets reçus sont mis dans une file d'attente jusqu'à la réception de tous les paquets. Ensuite, la BS commence la procédure de décodage en appliquant une transformation inverse de la matrice de transfert puis en utilisant le décodeur en métrique rang.

Notre objectif dans cette partie est d'utiliser un code en métrique rang dans les sources pour réduire les erreurs lors de la transmission. Nous proposons d'utiliser un code en métrique rang pour corriger les erreurs injectées par l'attaquant.

Soit Z la matrice erreur de dimension $s\mathrm{N}' \times m$ sur $\mathbb{F}_q$. La matrice Z est composée de s sous matrices $\mathrm{Z}_{i=1:s}$, on a

$$rang(\mathrm{Z}) = rang(\mathrm{Z}_1) + rang \begin{bmatrix} \mathrm{Z}_2 \\ \vdots \\ \mathrm{Z}_s \end{bmatrix} - dim(\langle \mathrm{Z}_1 \rangle \cap \left\langle \begin{bmatrix} \mathrm{Z}_2 \\ \vdots \\ \mathrm{Z}_s \end{bmatrix} \right\rangle).$$

Nous pouvons déduire que si $rank(\mathrm{Z}) = r$ et Z_1 vérifie

$$\langle \mathrm{Z}_1 \rangle \cap \left\langle \begin{bmatrix} \mathrm{Z}_2 \\ \vdots \\ \mathrm{Z}_s \end{bmatrix} \right\rangle = \{0\}, \tag{5.15}$$

alors, le rang moyen de Z_1 est $\frac{r}{s}$. Cependant, ce cas se produit avec une faible probabilité. En général, puisque les combinaisons se font d'une manière aléatoire, l'intersection (5.15) n'est pas $\{0\}$. Par conséquent, le rang moyen de Z_1 est plus grand que$\frac{r}{s}$, ce qui implique que les performances de décodage se détériorent. L'idée principal pour garantir un décodage optimal des paquets sources est d'utiliser l'information de l'erreur qui est dans les erreurs des autres sources. Pour cette raison, nous avons proposé l'algorithme de décodage des codes LRPC modifiés(M-LRPC). Le principe de cet algorithme est d'utiliser l'information fournie par les syndromes des autres utilisateurs. Pour cela, on pré-calcule la base de l'erreur à partir de tous les utilisateurs. Ensuite, on applique le décodeur LRPC. La complexité de calcul des syndromes est $\mathcal{O}(s\mathrm{N}m(\mathrm{N}-k))$ multiplication dans $\mathbb{F}_q$, qui est négligeable en comparaison avec la complexité de l'algorithme de décodage du code LRPC.

La figure 3 montre un exemple de décodage à la station de base utilisant l'algorithme M-LRPC. Les matrices $Y_{i=1,2,3}$ représentent les paquets envoyés par les trois sources, elle sont de taille $N \times m$. Chaque ligne de ces matrices est une combinaisons des paquets codés par les sources. On commence par l'application de la méthode décrite dans 5.11 pour chaque matrice de transfert utilisée dans chaque transmission afin de récupérer les paquets codés. On applique la permutation inverse dans le but de mettre en ordre les paquets de chaque source. Ensuite, on calcule l'espace S généré par les coefficients syndrome de tous les sources. On suppose que la dimension de S est égale à rd. Le support de l'erreur est donné par :

$$E = S_1 \cap S_2 \cap ... \cap S_d,$$

où $S_i = F_i^{-1}S$ pour $i = 1 : d$.

L'étape finale consiste à utiliser le décodeur LRPC dans (5.5) pour pouvoir récupérer les paquets de chaque source.

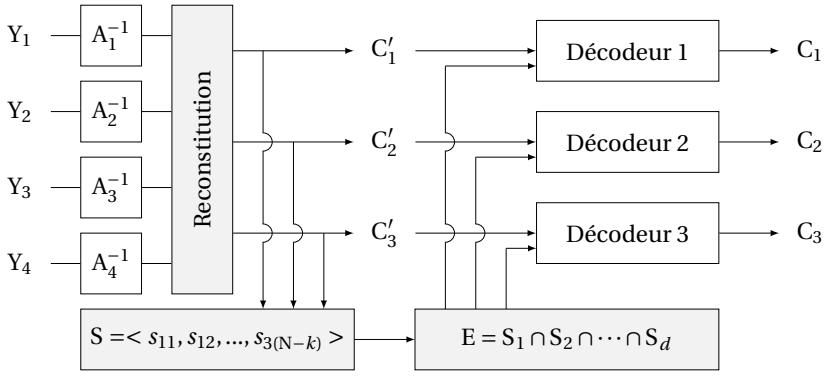


FIGURE 3 – Exemple de structure de décodage à la station de base utilisant l'algorithme M-LRPC.

Dans ce qui suit, nous proposons l'algorithme M-LRPC pour récupérer les paquets sources. Plus tard, nous présentons les probabilités d'échec de décodage de cet algorithme.

Algorithme 8 Algorithme de décodage M-LRPC

Entrée:

$\mathbf{H}$, ▷ la matrice de parité

C', ▷ le mot reçu

d, ▷ le poids de $\mathbf{H}$

E, ▷ la base de l'erreur e

Sortie:

x, ▷ le message

1: $s \leftarrow \mathbf{H}.C'^T$

2: $\{E_1, E_2, \ldots, E_r\} \leftarrow \mathbf{basis}(E)$

3: $s' \leftarrow (s_{111}, ..., s_{11r}, ..., s_{(n-k)dr})$

4: $e' \leftarrow \mathbf{Resolve}(A^r_{\mathbf{H}}, s')$ ▷ $A^r_H.e' = s'$

5: $(e_{11}, e_{12}, ..., e_{nr}) \leftarrow e'$
6: **pour** $i := 1$ **to** n **faire**
7: $\quad e_i \leftarrow \sum_{j=1}^{r} e_{ij} \mathrm{E}_j$
8: **fin pour**
9: $x \leftarrow$ **Resolve**$(\mathrm{G}, y - e)$ $\quad \triangleright\ x.\mathrm{G} = \mathrm{C}' - e$

Notons que l'algorithme 8 permet de décoder les paquets reçus puisque on a supposé que $dim(\mathrm{S})$=rd. La proposition suivante présente la probabilité de succès de l'algorithme de décodage M-LRPC.

Proposition 5.5. *La probabilité que l'algorithme de décodage M-LRPC réussisse à récupérer le message initial vérifie*

$$\mathrm{P_S}\,(k, \mathrm{N}, r) = \prod_{j=0}^{dr-1} \left(1 - q^{j-(\mathrm{N}-k-t)}\right) \times \prod_{j=0}^{dr-1} \left(1 - q^{j-m}\right) \times \prod_{j=0}^{r(d^2-d-1)-1} \left(1 - q^{j-m}\right). \tag{5.16}$$

Démonstration. En suivant exactement le même raisonnement dans la démonstration de le théorème 3.2 du chapitre 3 on obtient le résultat. Pour cela, il suffit de remplacer t par 0 dans la formule. □

Deux codes Gabidulin ne peuvent pas être superposés. En effet, la structure du code Gabidulin qui utilise les q-polynômes ne lui permet pas de faire la superposition de deux mots de codes ni des matrices de parité des deux codes. Contrairement au code Gabidulin, les codes LRPC peuvent être superposés. Cela est possible grâce à la manière dont les matrices de parité sont construites. En effet, les matrices ont une forme aléatoire, la concaténation de deux matrices de parité de deux codes LRPC donne une matrice de parité LRPC à condition que l'espace F soit le même pour les deux codes pour avoir un code du même paramètre d. Si l'espace F n'est pas le même, le code généré est un code LRPC de paramètre $\tilde{d} = dim\,(\mathrm{F}_1 + \mathrm{F}_2)$.

Maintenant, nous nous intéressons à l'approximation de la probabilité $\mathrm{P_R}$ que l'algorithme M-LRPC ne récupère pas les paquets transmis dans le cas de la présence de r paquets injectés dans le réseau. Soit C' un matrice de taille $sm \times \mathrm{N}$ construite par la superposition de s matrices $\mathrm{C}'_1, \mathrm{C}'_2, ..., \mathrm{C}'_s$, exprimées comme suit

$$\mathrm{C}'_i = \mathrm{A}_i^{-1} \times \mathrm{Y}_i\ ,$$

où $\mathrm{A}_1, \mathrm{A}_2, ..., \mathrm{A}_s$ sont les patrices de transfert de taille $m \times m$ de rang plein.

Lemme 5.1. *On a* $rank(\mathrm{C}') = rank(\mathrm{Y})$

Démonstration. La matrice C' peut être exprimée sous la forme du produit de la matrice Y où

$$\mathrm{A} = \begin{pmatrix} \mathrm{A}_1^{-1} & & & \\ & \mathrm{A}_2^{-1} & & \\ & & \ddots & \\ & & & \mathrm{A}_s^{-1} \end{pmatrix}.$$

A est de rang plein, Donc $rank(\mathrm{C}') = rank(\mathrm{A} \times \mathrm{Y}) = rank(\mathrm{Y})$. □

Le modèle proposé peut être vu comme un réseau à une source où la matrice de parité est la superposition des matrices de parité des s sources, et la matrice de transfert et A (*coding-independently decoding-jointly*). Cette méthode donne les même performances que l'algorithme M-LRPC en terme de décodage mais elle reste plus complexe.

Proposition 5.6. *La probabilité que la matrice* C' *soit de rang* r *est définie par*

$$\mathbb{P}(rank(\mathrm{C}') = r) = \frac{\mathrm{S}(m,r,q,r)}{q^{mr}}.$$

Démonstration. Ce théorème est déduit des équations 2.3 et 5.1. Soit Y' une matrice composée de r lignes non nulles. La probabilité que cette matrice soit de rang r est $\frac{\mathrm{S}(m,r,q,r)}{q^{mr}}$. D'autre part, $rank(\mathrm{C}') = rank(\mathrm{Y}) = rank(\mathrm{Y}')$. Donc, $\mathbb{P}(rank(\mathrm{C}') = r)$ est égale à $\mathbb{P}(rank(\mathrm{Y}') = r)$. □

Ce théorème sera utilisé pour dériver la probabilité d'échec de décodage $\mathrm{P_R}$. Maintenant, on considère le modèle de système décrit précédemment, on établit le théorème suivant.

Théorème 5.3. *La probabilité d'échec de décodage de l'algorithme M-LRPC pour un réseau multi-source utilisant le codage réseau aléatoire pour les paramètres* q, m, N, s *et* r*, est donnée par*

$$\mathrm{P_R}(r) = \sum_{j=0}^{r} \frac{\mathrm{S}(m,r,q,j)}{q^{mr}} \left(1 - \mathrm{P_S}\left(sk, s\mathrm{N}, j\right)\right).$$

Démonstration. Supposons que la station de base reçoit C' et notons $\mathrm{E} = \mathrm{C}' - \mathrm{C}$. Notons $\mathrm{C}' \neq \mathrm{C} \mid rank(\mathrm{E}) = r$ l'échec de décodage de C' sachant que le rang de E est r. Utilisons *la formule des probabilités totales*, on obtient la formule suivante de probabilité d'échec de décodage

$$\mathrm{P_R}(r) = \sum_{j=0}^{r} \mathbb{P}(rank(\mathrm{E}) = j)\mathbb{P}(\mathrm{C}' \neq \mathrm{C} \mid rank(\mathrm{E}) = j).$$

d'après la proposition 5.5 et la proposition 5.6, on déduit le résultat. □

5.3 Résultats numériques

Dans cette section, nous étudions les performances du modèle proposé à travers des simulations et nous les comparons avec les résultats des codes Gabidulin.

On fixe le nombre des paquets de chaque source à $m = 40$ et le nombre de nœuds source à $s = 1$, $s = 5$ et $s = 10$. Chaque source encode les m-messages de longueurs $k = 15$ avec un code en métrique rang de paramètres $[\mathrm{N}, k, d]$ (code LRPC et code Gabidulin) où $\mathrm{N} = 30$ et $d = 2$. On utilise arbitrairement une modulation par changement de phase binaire (BPSK) et on simule le scénario où tous les nœuds utilisent un CRC pour détecter les erreurs du canal.

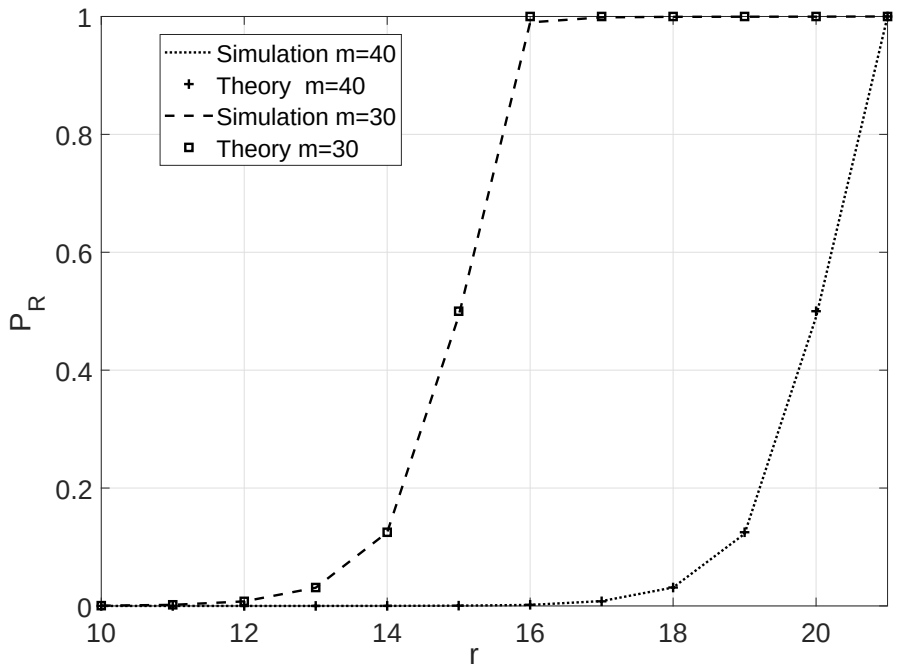


FIGURE 4 – Les performances de M-LRPC dans un réseau multi-sources en fonction de r, pour $s = 10$ et différentes valeurs de m.

Nous commençons par illustrer l'exactitude de la formule analytique de la probabilité de décodage. A cette fin, nous considérons le modèle du réseau multi-sources décrit dans la section précédente. L'analyse analytique du modèle étudié joue un rôle important dans l'étude des performances du code dans le cas de changement des paramètres, du canal ou du modèle lui même.

Pour simplifier le modèle, nous supposons que r paquets sont injectés dans le réseau, nous supposons de plus que la représentation matricielle de l'erreur est de rang r. nous utilisons la formule de probabilité d'échec de décodage exprimée dans le théorème 5.3 et nous comparons le résultat avec le résultat de simulation.

La Figure 4 décrit le résultat de simulation et l'expression théorique de la probabilité d'échec de décodage P_R de l'algorithme de décodage M-LRPC pour $m = 30$ et $m = 40$ en fonction du nombre d'erreurs injectées dans le réseau pour $s = 10$ sources. On remarque que les résultats de simulations sont très proches des résultats théoriques données dans le théorème 5.3.

La Figure 5 illustre la probabilité d'échec de décodage P_R en fonction de r pour différentes valeurs de nombre de sources et la taille de corps fixée à $m = 40$. On remarque que le M-LRPC a un bon comportement par rapport au code Gabidulin pour $s = 5$ et $s = 10$. On peut prédire ce résultat puisque le M-LRPC utilise l'information de l'erreur des autres sources dans l'algorithme de décodage. La capacité de correction du code Gabidulin est $\frac{N-k}{2}$. Si le nombre des erreurs r est supérieure à cette valeur, la station de base ne peut pas récupérer le message transmit par la source et les sources doivent renvoyer tous les paquets. D'autre part, la capacité de correction de M-LRPC est égale

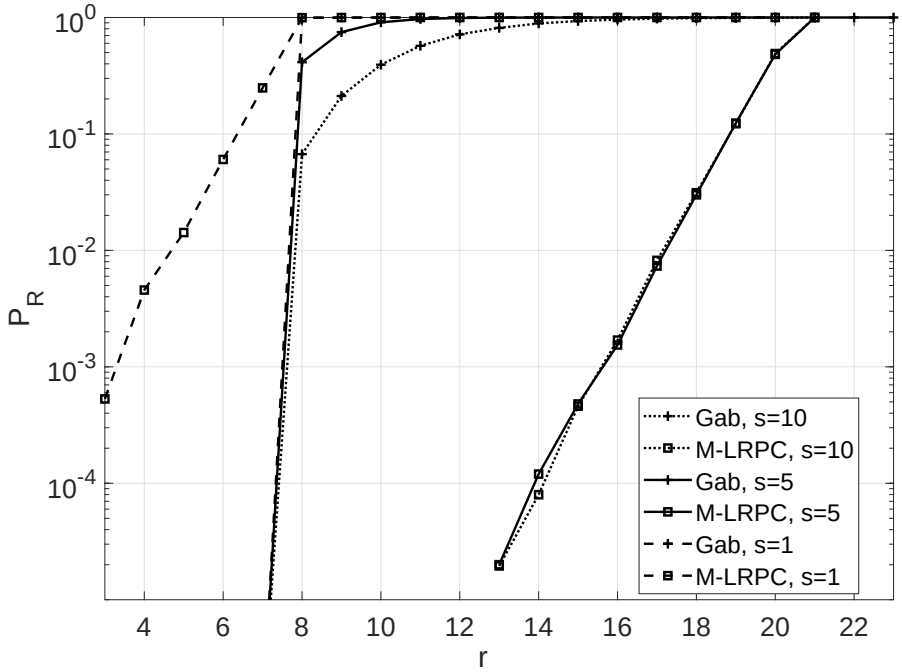


FIGURE 5 – Les performances de M-LRPC comparées au code Gabidulin dans un réseau multi-sources en fonction de r, pour $m = 40$ et différentes valeurs de s.

à $min(m, s(\mathrm{N} - k))/d$ au maximum, qui est 40 quand $s = 5$ ou $s = 10$, et 7 quand $s = 1$. Notons que dans le cas où $s = 5$ or $s = 10$, la probabilité de décodage augmente pour M-LRPC. Les performances du décodage séparé des codes Gabidulin sont affectés, ce qui affecte les performances de tous le système.

On remarque que les codes Gabidulin sont plus performants que les M-LRPC qui se comportent comme des codes LRPC dans le cas $s = 1$. La capacité de décodage dans ce cas vaut 7 pour les codes Gabidulin et vaut 7 au maximum pour les codes LRPC.

Finalement, pour exploiter l'avantage de l'utilisation d'un corps de base de taille grande, on prend $m = 30$ où la capacité de décodage du code Gabidulin reste inchangée.

Les performances de M-LRPC dans le cas du codage réseau aléatoire sont étudiées pour $m = 30$ dans la Figure 6. Dans ce scénario, on compare les performances des codes Gabiduling et M-LRPC pour les mêmes valeurs $s = 1$, $s = 5$ et $s = 10$. On remarque que la capacité de correction M-LRPC se détériore lorsque $s = 5$ et $s = 10$, pendant qu'elle reste inchangée pour $s = 1$. Ceci peut être expliqué par le fait que la probabilité d'échec de décodage $\mathrm{P_R}$ dépend de m pour $s = 5$ et $s = 10$. Pour la même raison, on remarque qu'il n'y a pas de détérioration dans les performances des codes Gabidulin. Il est évident, d'après la Figure 6, que le modèle proposé reste plus performant par rapport aux codes Gabidulin.

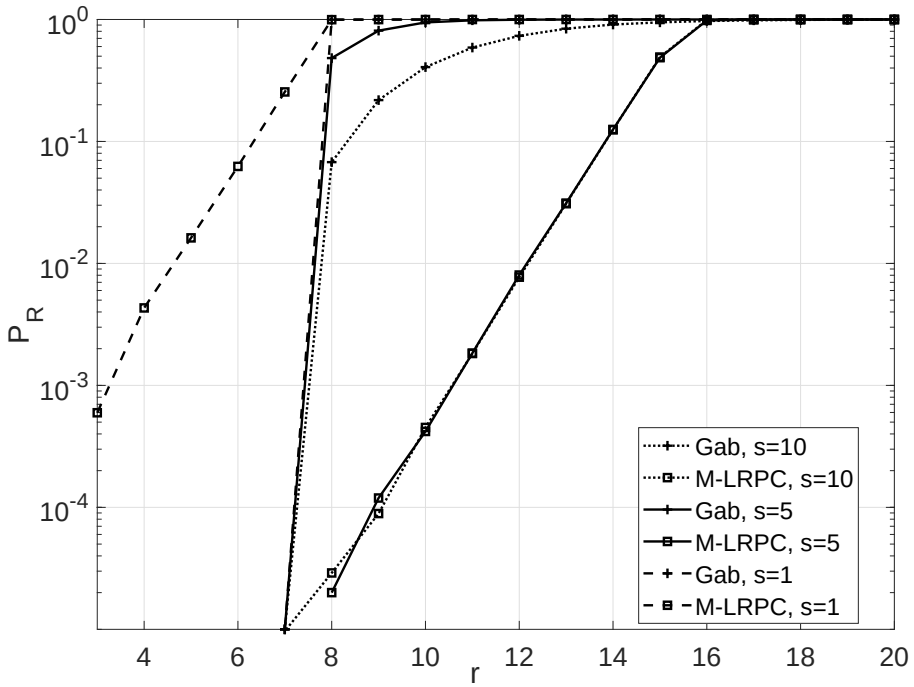


FIGURE 6 – Les performances de M-LRPC comparées au code Gabidulin dans un réseau multi-sources en fonction de r, pour $m = 30$ et différentes valeurs de s.

5.4 Conclusion

Dans ce chapitre, avons considéré un scénario de collecte de données émises par plusieurs sources vers une station de base. Nous avons considéré deux cas de codage réseau dans un scénario de collecte de données. Le premier cas est le codage réseau cohérent où la topologie du réseau ainsi que la matrice de transfert sont supposées être connues par la station de base. Le second cas est le codage réseau non cohérent. Dans ce cas la station de base n'a aucune connaissance sur la topologie du réseau.

Dans la deuxième partie de ce chapitre, nous avons adapté les codes LRPC au réseau multi-sources. Dans ce contexte, nous avons proposé l'algorithme de décodage M-LRPC qui est basé sur l'algorithme de décodage des codes LRPC classique. Ensuite, nous avons dérivé l'expression analytique de l'algorithme de décodage M-LRPC pour le modèle considéré. Nous avons prouvé, à travers des simulations, que les codes LRPC sont plus efficaces dans les réseaux multi-sources que dans un réseau à une seule source grâce à leur structure. Nous avons également montré que les codes LRPC résistent aux effacements contrairement aux codes Gabidulin.

Chapitre 6

Conclusion et perspectives

Nous avons étudié, dans ce rapport de thèse, l'optimisation des transmissions dans les réseaux de capteurs sans fil. Deux domaines se sont dégagés et ont été présentés dans cette thèse. Le premier est la théorie des codes en métrique rang et le deuxième est le codage réseau. Des travaux ont montré que la capacité maximale de multicast peut être atteinte en utilisant le codage réseau (voir chapitre 4). Cependant, l'utilisation du codage réseau augmente le risque de la propagation d'erreurs. Pour remédier à ce genre de problème, nous avons proposé les codes en métrique rang. Nous avons adapté ces codes, qui ont été introduits dans le contexte de la cryptographie, à la correction d'erreurs. Pour cela, nous avons étudié le comportement de deux familles de codes dans un scénario de transmission sans fil utilisant le codage réseau. La première est la famille des codes Gabidulin et la deuxième est la famille des codes LRPC. Dans ce contexte, nous avons considéré trois types d'erreurs, le bruit de fond, les erreurs injectés dans le réseau par un utilisateur malveillant et les effacements qui peuvent être dus aux pannes des nœuds.

L'analyse que nous avons faite sur la famille des codes LRPC a montré que ces codes sont plus adaptés aux réseaux de capteurs sans fil que les codes Gabidulin utilisés dans la littérature. Nous avons également démontré que, pour certains paramètres du code, un code LRPC peut atteindre la borne de Gilbert-Varshamov en métrique rang. De plus, nous avons pu adapter les codes LRPC dans le contexte du codage réseau aléatoire pour un réseau multi-sources. Les résultats que nous avons trouvés sont aussi intéressants qu'inattendus. En effet, l'utilisation des codes LRPC dans un réseau multi-sources permet d'avoir de meilleures performances, en terme de capacité de correction, par rapport à un réseau à une seule source. Les résultats de cette thèse sont résumés dans la suite.

Le chapitre 3 est dédié aux codes LRPC. Premièrement, nous avons introduit les codes LRPC matriciels et nous avons donné leurs principales propriétés. Nous avons étudié asymptotiquement la capacité de correction de cette famille de codes pour différents paramètres. Nous avons montré que pour certains paramètres les codes LRPC peuvent atteindre la borne Gilbert-Varshamov en métrique rang. Ensuite, nous avons présenté les codes LRPC étendus. Par ailleurs, nous avons proposé une analyse théorique de la probabilité de décodage pour ces types de codes et nous avons montré

à travers des simulations que la probabilité de décodage calculée analytiquement pour les codes LRPC est exacte.
Dans le chapitre 4, nous avons étudié le problème de collecte de données dans les réseau de capteurs sans fil et nous avons proposé un modèle de codage de données basé sur la concaténation d'un code LRPC avec un code Convolutif. La méthode proposée tient compte des erreurs introduites par un utilisateur malveillant et des erreurs de propagation. Nous avons donné l'expression de la probabilité du décodage des codes Gabidulin et des codes LRPC dans le cas du codage réseau aléatoire. Les résultats de simulation ont montré que le modèle proposé est plus performant que le modèle basé sur la concaténation d'un code Gabidulin et d'un code Reed-Solomon, utilisé dans la littérature. Nous avons également montré que les résultats théoriques sont très proches des résultats de simulation.
Finalement, dans le chapitre 5 nous avons étudié un scénario de transmission dans un réseau multi-sources. Dans ce contexte, nous avons proposé l'algorithme de décodage M-LRPC et nous avons prouvé qu'ils sont plus efficaces dans les réseaux multi-sources que dans un réseau à une seule source grâce à leur structure. Nous avons également montré que les codes LRPC résistent aux effacements contrairement aux codes Gabidulin.

Ces résultats nous ont motivé pour penser à une implémentation matérielle des codes LRPC dans un contexte multi-sources. Cependant, la solution proposée reste plutôt théorique et difficilement applicable dans le cas d'une gestion décentralisée du réseau. En effet, dans cette thèse, nous avons considéré une concaténation de s matrices identité avec le mot de code, où s est le nombre des sources. Si plusieurs sources décident de transmettre des paquets, le rendement du code dans ce cas est très faible et la capacité de correction est très petite par rapport à la taille des paquets codés. De plus, la taille des paquets codés est très grande par rapport à la taille du message initial, ce qui rend l'utilisation de cette technique dans un réseau de capteurs très difficile. Pour remédier à ce problème, on suppose qu'il y a une phase d'initialisation dans laquelle les sources demandent les numéros des entêtes, à la station de base, pour éviter les collisions. En effet, si deux sources choisissent les mêmes positions pour coder des paquets, la destination ne pourra pas les décoder. Par conséquent, les deux sources doivent choisir des positions différentes pour chaque paquet. Cette solution parait envisageable, cependant, dans les scénarios où la station de base ne peut pas échanger avec les utilisateurs, ces derniers choisissent aléatoirement les positions des paquets codés et cela augmente le risque de la propagation des erreurs. Pour résoudre ce problème, on pourrait étudier des algorithmes de décodage adaptés au contexte de transmissions multi-sources dans les réseaux de capteurs sans fil. Le but serait d'obtenir un remplissage de trame plus efficace qui puisse garantir un taux de codage et un rendement compétitif par rapport aux techniques de type RLNC.

Bibliographie

[Ace+05] Szymon ACEDANSKI et al. « How good is random linear coding based distributed networked storage ». In : *Workshop on Network Coding, Theory and Applications*. 2005, p. 1-6 (cf. p. 9).

[Ahl+00] Rudolf AHLSWEDE et al. « Network information flow ». In : *IEEE Transactions on information theory* 46.4 (2000), p. 1204-1216 (cf. p. 8, 55).

[Aky+02] Ian F AKYILDIZ et al. « Wireless sensor networks : a survey ». In : *Computer networks* 38.4 (2002), p. 393-422 (cf. p. 8).

[Bas+13] Riccardo BASSOLI et al. « Network coding theory : A survey ». In : *IEEE Communications Surveys & Tutorials* 15.4 (2013), p. 1950-1978 (cf. p. 60-62).

[Ber84] E. R. BERLEKAMP. « Algebraic Coding Theory, revised ed. Aegean Park Press ». In : (1984) (cf. p. 17).

[CY02] Ning CAI et Raymond W YEUNG. « Network coding and error correction ». In : *Information Theory Workshop, 2002. Proceedings of the 2002 IEEE*. IEEE. 2002, p. 119-122 (cf. p. 13).

[CY+06] Ning CAI, Raymond W YEUNG et al. « Network error correction, II : Lower bounds ». In : *Communications in Information & Systems* 6.1 (2006), p. 37-54 (cf. p. 13, 88).

[Del78] Ph DELSARTE. « Bilinear forms over a finite field, with applications to coding theory ». In : *Journal of Combinatorial Theory, Series A* 25.3 (1978), p. 226-241 (cf. p. 17, 27).

[Fan+14] Yang FANGCHUN et al. « An overview of internet of vehicles ». In : *China Communications* 11.10 (2014), p. 1-15 (cf. p. 7).

[FS+08] Christina FRAGOULI, Emina SOLJANIN et al. « Network coding applications ». In : *Foundations and Trends® in Networking* 2.2 (2008), p. 135-269 (cf. p. 9).

[Gab+13] Philippe GABORIT et al. « Low rank parity check codes and their application to cryptography ». In : *Proceedings of the Workshop on Coding and Cryptography WCC*. T. 2013. 2013 (cf. p. 14, 37, 38).

[Gab85] Ernest Mukhamedovich GABIDULIN. « Theory of codes with maximum rank distance ». In : *Problemy Peredachi Informatsii* 21.1 (1985), p. 3-16 (cf. p. 17, 27, 28).

[GY06] Maximilien GADOULEAU et Zhiyuan YAN. « GENp1-1 : Properties of Codes with the Rank Metric ». In : *IEEE Globecom 2006*. IEEE. 2006, p. 1-5 (cf. p. 27).

[Ho+03] Tracey HO et al. « The benefits of coding over routing in a randomized setting ». In : (2003) (cf. p. 10, 55, 64, 65).

[Ho+06] Tracey HO et al. « A random linear network coding approach to multicast ». In : *IEEE Transactions on Information Theory* 52.10 (2006), p. 4413-4430 (cf. p. 10, 13, 55, 64).

[KK08] Ralf KOETTER et Frank R KSCHISCHANG. « Coding for errors and erasures in random network coding ». In : *IEEE Transactions on Information theory* 54.8 (2008), p. 3579-3591 (cf. p. 10, 13).

[KM03] Ralf KOETTER et Muriel MÉDARD. « An algebraic approach to network coding ». In : *IEEE/ACM Transactions on Networking (TON)* 11.5 (2003), p. 782-795 (cf. p. 60, 61).

[LNC83] Rudolf LIDL, Harald NIEDERREITER et P. M. COHN. « Encyclopedia of mathematics and its applications vol 20 ». In : (1983) (cf. p. 17, 21).

[Loi04] P LOIDREAU. « On the reconstruction of linearized polynomials : a new decoding algorithm for Gabidulin codes ». In : *COMPTES RENDUS MATHEMATIQUE* 339.10 (2004), p. 745-750 (cf. p. 30).

[Loi14] Pierre LOIDREAU. « Asymptotic behaviour of codes in rank metric over finite fields ». In : *Designs, codes and cryptography* 71.1 (2014), p. 105-118 (cf. p. 28).

[Lub02] Michael LUBY. « LT codes ». In : *null*. IEEE. 2002, p. 271 (cf. p. 66).

[Lub+06] Michael LUBY et al. « Raptor codes for reliable download delivery in wireless broadcast systems ». In : *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006*. T. 1. IEEE. 2006, p. 192-197 (cf. p. 66).

[LYC03] S-YR LI, Raymond W YEUNG et Ning CAI. « Linear network coding ». In : *IEEE transactions on information theory* 49.2 (2003), p. 371-381 (cf. p. 58, 60).

[Mat07] Ryutaroh MATSUMOTO. « Construction algorithm for network error-correcting codes attaining the singleton bound ». In : *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 90.9 (2007), p. 1729-1735 (cf. p. 88).

[Mon+07] Gabriel MONTENEGRO et al. *Transmission of IPv6 packets over IEEE 802.15. 4 networks*. Rapp. tech. 2007 (cf. p. 8).

[Ore33] Oystein ORE. « On a special class of polynomials ». In : *Transactions of the American Mathematical Society* 35.3 (1933), p. 559-584 (cf. p. 18, 20).

[PS98] Christos H PAPADIMITRIOU et Kenneth STEIGLITZ. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1998 (cf. p. 56).

[Rot91] Ron M. ROTH. « Maximum-rank array codes and their application to crisscross error correction ». In : *IEEE transactions on Information Theory* 37.2 (1991), p. 328-336 (cf. p. 17).

[RP04] Gerd RICHTER et Simon PLASS. « Error and erasure decoding of rank-codes with a modified Berlekamp-Massey algorithm ». In : *ITG FACHBERICHT* (2004), p. 203-210 (cf. p. 32-35).

[Sil09] Danilo SILVA. « Error control for network coding ». Thèse de doct. 2009 (cf. p. 85-87, 89, 90).

[SKK08] Danilo SILVA, Frank R KSCHISCHANG et Ralf KOETTER. « A rank-metric approach to error control in random network coding ». In : *IEEE transactions on information theory* 54.9 (2008), p. 3951-3967 (cf. p. 10, 13, 85, 88).

[SRV12] Natalia SILBERSTEIN, Ankit Singh RAWAT et Sriram VISHWANATH. « Error resilience in distributed storage via rank-metric codes ». In : *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2012, p. 1150-1157 (cf. p. 78).

[TCBOF11] Óscar TRULLOLS CRUCES, José María BARCELÓ ORDINAS et Marco FIORE. « Exact decoding probability under random linear network coding ». In : *IEEE communications letters* 15.1 (2011), p. 67-69 (cf. p. 69).

[YC+06] Raymond W YEUNG, Ning CAI et al. « Network error correction, I : Basic concepts and upper bounds ». In : *Communications in Information & Systems* 6.1 (2006), p. 19-35 (cf. p. 88).

[Yeu+05] RW YEUNG et al. *Network Coding Theory, Foundation and Trends in Communications and Information Technology*. 2005 (cf. p. 13).

[YZ99] Raymond W YEUNG et Zhen ZHANG. « Distributed source coding for satellite communications ». In : *IEEE Transactions on Information Theory* 45.4 (1999), p. 1111-1120 (cf. p. 8, 9).

[Zha12] Xubo ZHAO. « Notes on" exact decoding probability under random linear network coding" ». In : *IEEE Communications Letters* 16.5 (2012), p. 720-721 (cf. p. 69).

Optimisation des codes en métrique rang pour les systèmes de communication sans fil

Résumé — Dans cette thèse, nous avons envisagé l'utilisation des codes en métrique rang pour des applications de communication sans fil en général, et les réseaux de capteurs en particulier. Après avoir introduit les codes en métrique rang, ces codes, qui ont été proposés dans le contexte de la cryptographie, sont adaptés par la suite pour la correction d'erreurs. Pour cela, une étude est faite sur le comportement de ces familles de codes dans un scénario de transmission sans fil en utilisant le codage réseau. Dans ce contexte, trois types d'erreurs sont considérés : le bruit de fond, les erreurs injectés dans le réseau par un utilisateur malveillant et les effacements qui peuvent être dus aux pannes des nœuds. L'analyse qui a été faite sur la famille des codes Low Rank Parity Check (LRPC) a montré que ces derniers sont plus adaptés aux réseaux de capteurs sans fil par rapport aux codes Gabidulin utilisés dans la littérature. Cette analyse a été généralisée dans le contexte multi-sources et a montré que les codes LRPC sont plus efficaces dans ce contexte. Ces contributions apportent un nouveau souffle à l'utilisation des codes en métrique rang et offrent des perspectives de poursuite intéressantes.

Mots clés : Codes en métrique rang, codage réseau, codes LRPC, réseau de capteurs sans fil.

Optimization of rank metric codes for wireless communication systems

Abstract — In this thesis, we have considered the rank metric codes for wireless sensor networks. Firstly, we have introduced the rank metric codes. Then, we adapted these codes, which were originally dedicated to cryptography applications, for error correction. To this end, we have studied the behavior of the family of rank metric codes in a wireless communication scenario using network coding. In this context, three types of errors are considered, background noise, errors injected into the network by a malicious user and erasures caused by node failures. Our analysis of the Low Rank Parity Check codes (LRPC) has shown that they are more suited to wireless sensor networks and they perform better than Gabidulin codes used in the literature. This analysis has been generalized in the multi-source context and has shown that LRPC codes are more efficient in this context compared to Gabidulin codes. These contributions give a new incentive for the use of rank metric codes and offer interesting perspectives.

Keywords : Rank metric codes, network coding, LRPC codes, wireless sensor network.

XLIM - UMR CNRS n° 7252
123 Avenue Albert Thomas, 87060 LIMOGES Cedex